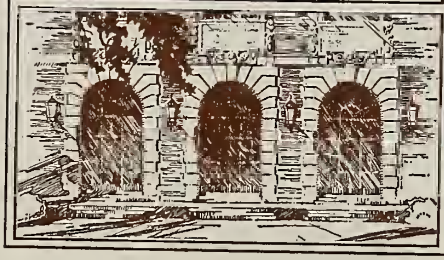


math

UIUCDCS-R-75-714

PROGRAM MANUAL:
NOR NETWORK TRANSDUCTION BY GENERALIZED GATE MERGING AND SUBSTITUTION
(Reference Manual of NOR Network Transduction Programs NETTRA-G3 and NETTRA-G4)

by

H.C. Lai

April 1975

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN · URBANA, ILLINOIS

UIUCDCS-R-75-714

PROGRAM MANUAL:
NOR NETWORK TRANSDUCTION BY GENERALIZED GATE MERGING AND SUBSTITUTION

(Reference Manual of NOR Network Transduction
Programs NETTRA-G3 and NETTRA-G4)

by

H.C. Lai

April 1975

Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

This work was supported in part by the National Science Foundation under Grant No. GJ-40221.

ABSTRACT

This paper explains the operation and usage of two FORTRAN programs NETTRA-G3 and NETTRA-G4, developed for NOR-network transduction (transformation and reduction).

Existing (non-optimal) NOR-gate networks and their required output functions are given to the programs as input. Program NETTRA-G3 reduces the number of gates in the network by means of merging gates. When two gates are merged to one, the merged gate will be connected to the immediate successors of the two gates, and the two gates will be removed from the network. On the other hand, program NETTRA-G4 reduces the number of gates by means of substituting for all output corrections of a particular gate under consideration. If these connections can be successfully replaced, this particular gate then can be removed from the network.

These programs are only two out of a whole system of programs, designated by the name "NETTRA" (for NETwork TRAnsduction), which implement different NOR-network transduction procedures.

The theoretical basis for the algorithms implemented by NETTRA-G3 and -G4 is detailed in earlier reports ([1] and [3]).

ACKNOWLEDGMENT

The author is greatly appreciative of Professor S. Muroga for his discussions and guidance relating to the preparation of this paper, and also, for his careful reading and valuable suggestions for the improvement of the original manuscript. The author is also indebted to J. N. Culliney and Y. Kambayashi upon whose related research much of the work reported herein depends.

This work was supported in part by the National Science Foundation under Grant No. GJ-40221.

TABLE OF CONTENTS

1. INTRODUCTION

This manual is intended to instruct the reader in the use of the FORTRAN programs NETTRA-G3 and NETTRA-G4. These programs realize the algorithms described in detail in [1].

NETTRA-G3 and -G4 are only two out of the whole system of programs developed by the research group of Professor S. Muroga at the University of Illinois. The generic name 'NETTRA' (for NETwork TRAnsduction) designates the whole collection of programs comprising the system. All of the programs in the NETTRA system either transform or assist in transforming and reducing a large, non-optimal network of NOR gates realizing one or more functions into a smaller, less expensive (in terms of the number of required gates and connections, for example), near-optimal network realizing the same function(s). In general, such a transduction could involve a complete reorganization of the network: the addition and/or deletion of gates; the addition and/or deletion of connections among gates; and/or the substitution of certain connections for others. The transduction procedures realized by NETTRA-G3 and -G4 actually involve these alterations with the exception of adding gates to the network.

The procedures realized in these programs NETTRA-G3 and -G4 aim at reducing the number of NOR gates in the network by merging gates and substituting for gates, respectively. They are more complex than those appearing in programs NETTRA-PG1, -P1 and -P2 and require more computer time to execute. However, they are more powerful also, and they can often reduce a network when it is impossible to do so by the procedures in

NETTRA-PG1, -P1, and -P2. The programs NETTRA-PG1, -P1, and -P2 described in [2], though, are more efficient than NETTRA-G3 and -G4 when first applied to large, far from optimal networks. NETTRA-G3 and -G4 are most useful when applied to more nearly optimal networks which is fairly difficult to be further reduced.

The next two sections, Sections 2 and 3, will discuss the two programs in great detail and present some examples of the effectiveness of their transductions. This will be followed, in Section 4, by a description of the functions of subroutines which support the subroutines actually realizing the procedures. Section 5 outlines the preparation of input for these programs. Finally, in the Appendix, a listing of all of the FORTRAN programs NETTRA-G3 and NETTRA-G4 will be given.

2. A NETWORK TRANSDUCTION PROCEDURE BY MERGING OF GATES

This section will discuss the NOR-network transduction procedure realized by the FORTRAN program designated NETTRA-G3. This program realizes a procedure which merges two NOR gates at a time. In a given network, if two gates can be replaced by one gate with inputs from external variables and/or existing gates not fed by the two gates, they are said to be mergeable. The gate replacing those two gates is called the merged gate. The procedure realized by program NETTRA-G3 is the procedure to examine every pair of gates in a network to see if they are mergeable. For the sake of efficiency this procedure does not use the necessary and sufficient conditions for finding mergeable gates. Instead, the concept of compatible sets of permissible functions† (CSPF) is used.

The input to this program is a description of a particular NOR network under consideration. This description (explained in great detail in Section 5) consists of a set of various network parameters. The output of this program is a description of the "transduced" network (if a transduction, in this case the merging of gates, was possible).

The entire NETTRA-G3 program requires 127K bytes of core storage, about 42K bytes being occupied by the actual program instructions and about 85K by the stored data.

The gate merging procedure is realized by the FORTRAN subroutine GTMERG. This subroutine and the following support subroutines, written in FORTRAN IV for the IBM 360/75, constitute the program NETTRA-G3:

---

†The reader is assumed to be familiar with the definitions presented in [1], [3].

MAIN, GTMERG, MINI2, OUTPUT and SUBNET. Two system-supplied timing subroutines STIMZE and KTIMEZ are also assumed to be available, but if they are not, their use can be omitted from the program, or another suitable timing routine substituted, without changing the procedure itself. The functions of the support subroutines MAIN, OUTPUT, and SUBNET will be discussed in Section 4. The function of subroutine MINI2 is to calculate compatible sets of permissible functions for all the gates in a given network. It is explained in great detail in [4] and [2].

The general organization of the program NETTRA-G3 is shown in Fig. 2.1. An arrow from block i to block j represents the fact that the subroutine represented by block i calls the subroutine represented by block j.

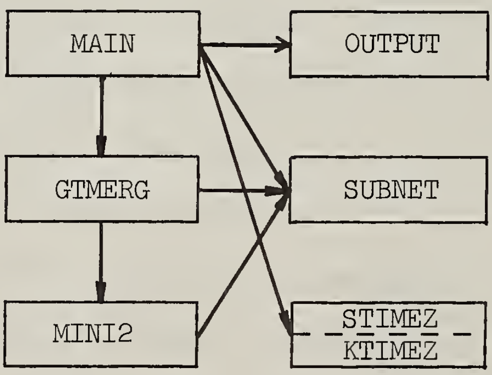


Fig. 2.1 General organization of the program NETTRA-G3.

Section 2.1 will describe the procedure itself, and Section 2.2 will show examples obtained by NETTRA-G3.

2.1 Gate Merging Procedure and Flowchart

The gate merging procedure is essentially performed by the subroutine GTMERG. GTMERG is programmed based on the procedure GMGC (a Generalized Merging of Gates using CSPF's) described in detail in [1]. The following discussions of GTMERG will assume knowledge of the information contained in [1].

The purposes of the main variables and arrays appearing in the subroutine will be explained in the program listing in the Appendix.

PROCEDURE FOR MERGING OF GATES

Step 1. Calculation of CSPF's

Call subroutine MINI2 to calculate a compatible set of permissible functions for every gate in the network.

Step 2. Select a Pair of Gates

Select two gates GI and GJ such that GJ > GI both according to the ascending order of gate labels.

If all possible combinations of gate pairs have been considered, return to the calling subroutine.

Step 3. Calculation of CSPF for the Merged Gate

Calculate the CSPF for the merged gate GIJ by forming the intersection of the two CSPF's for gates GI and GJ. If the CSPF for GIJ is a null set, GI and GJ are not mergeable, and go to Step 2.

Calculate the set of connectable functions for the merged gate GIJ.

Step 4. Check Substitutability

If the output of gate GI is included in the CSPF of gate GJ and gate GI is not a successor of gate GJ, then gate GJ can be replaced by gate GI, and go to step 8.

If the output of gate GJ is contained in the CSPF of gate GI, and gate GJ is not a successor of gate GI, then gate GI can be replaced by gate GJ. So interchange the labels of gates GI and GJ and go to step 8.

Step 5. Select Connectable Functions for Gate GIJ

Find all external variables and/or existing gates in the network which are contained in the set of connectable functions, and which are not fed by gate GI or GJ.

Step 6. Check Realizability of the Merged Gate

If connectable functions obtained in Step 5 do not realize a function contained in the CSPF of GIJ, go to step 2.

Step 7. Construct the Merged Gate

Disconnect all input connections of gate GI, connect all connectable functions found in Step 5 to gate GI to realize the merged gate at gate GI.

Step 8. Substitution

Disconnect all output connections of gate GJ, and connect gate GI to all immediate successors of gate GJ. The resulting network still realizes the specified function(s).

Step 9. Update Information on the New Network

Call subroutine SUBNET to update the information on the configuration of the network (e.g., predecessor lists and successor lists). Go to Step 1.

The flowchart of this procedure, realized by FORTRAN subroutine GTMER is shown in Fig. 2.2.

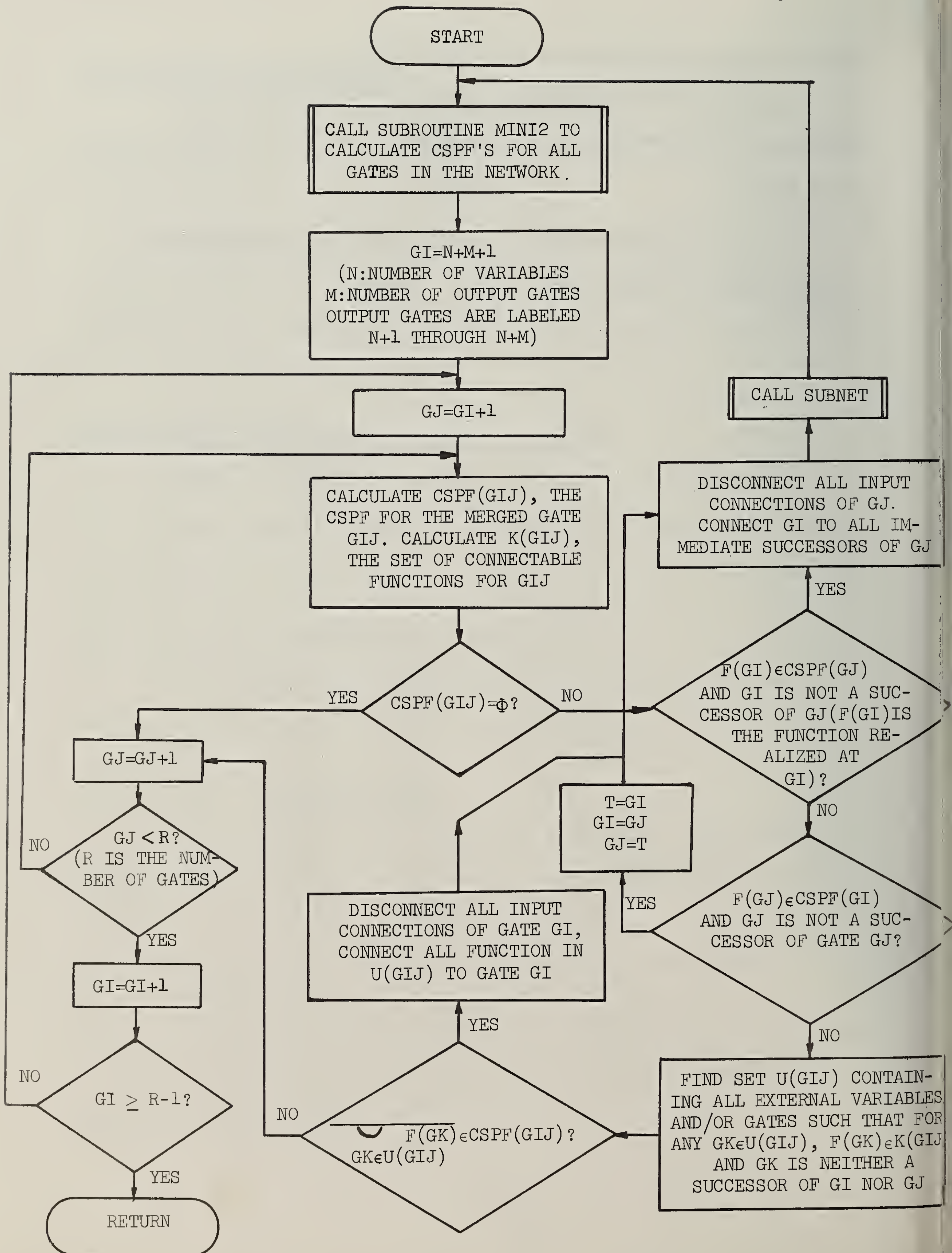


Fig. 2.2 Flowchart of subroutine GTMERG.

2.2 Examples for Program NETTRA-G3

The printout obtained during the solution of a typical problem by NETTRA-G3 is shown in Fig. 2.3. The original network, as specified in the beginning of the printout (Fig. 2.3 (a)), consists of 33 gates and 310 connections and realizes a single 5-variable output function. Only uncomplemented variables are assumed to be available as inputs to the network.

This information is followed by a complete truth table (b) showing the output of every gate in the original network for every possible input combinations. Note that it is gate 1 which realizes the output function of the network.

Next appears a description of the configuration of the network (c). Each gate is listed along with the gates and/or external variables which are its inputs. The level numbers, also to be seen in (c), will be discussed in Section 5.3.

The truth table (note that the outputs for disconnected gates are shown as all 1's) and network configuration for the transduced network resulting from the execution of NETTRA-G3 are shown in (d) and (e), respectively. The derived network, obtained in .74 seconds, consists of 12 gates and 38 connections.

```
**** 5 VAR. EXAMPLE                HEX=8B5809F0

      NUMBER OF VARIABLES =   5
      NUMBER OF FUNCTIONS =   1
      COST COEFFICIENT A  =1000
                       B  =   1

      --- UNCOMPLEMENTED VARIABLES X ---

ORIGINAL NETWORK   COST=33310
```

(a) Heading information and network parameter

Fig. 2.3 Printout obtained from NETTRA-G3 for a sample problem.

TRUTH TABLE

X1 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
X2 =	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
X3 =	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
X4 =	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
X5 =	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1 =	1	0	0	0	1	0	1	1	0	1	0	1	1	0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	0	0	0	0
2 =	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3 =	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4 =	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5 =	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6 =	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7 =	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8 =	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9 =	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10 =	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11 =	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12 =	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
13 =	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14 =	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15 =	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
21 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
22 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
23 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
24 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
25 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
26 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
27 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
28 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
29 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
30 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
31 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
32 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
33 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

(b) Truth table for original network.

GATE ..	LEVEL	FED BY
1	/ 1/	3 4 5 7 10 12 15 16 17 18 19 20 21 23 24 30 31 32 33
2	/ 7/	X1 X2 X3 X4 X5
3	/ 6/	X1 X2 X3 X4 2
4	/ 6/	X1 X2 X3 X5 2
5	/ 5/	X1 X2 X3 2 3 4
6	/ 6/	X1 X2 X4 X5 2
7	/ 5/	X1 X2 X4 2 3 6
8	/ 5/	X1 X2 X5 2 4 6
9	/ 4/	X1 X2 2 3 4 5 6 7 8
10	/ 6/	X1 X3 X4 X5 2
11	/ 5/	X1 X3 X4 2 3 10
12	/ 5/	X1 X3 X5 2 4 10
13	/ 4/	X1 X3 2 3 4 5 10 11 12
14	/ 5/	X1 X4 X5 2 6 10
15	/ 4/	X1 X4 2 3 6 7 10 11 14
16	/ 4/	X1 X5 2 4 6 8 10 12 14
17	/ 3/	X1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
18	/ 6/	X2 X3 X4 X5 2
19	/ 5/	X2 X3 X4 2 3 18
20	/ 5/	X2 X3 X5 2 4 18
21	/ 4/	X2 X3 2 3 4 5 18 19 20
22	/ 5/	X2 X4 X5 2 6 18
23	/ 4/	X2 X4 2 3 6 7 18 19 22
24	/ 4/	X2 X5 2 4 6 8 18 20 22
25	/ 3/	X2 2 3 4 5 6 7 8 9 18 19 20 21 22 23 24
26	/ 5/	X3 X4 X5 2 10 18
27	/ 4/	X3 X4 2 3 10 11 18 19 26
28	/ 4/	X3 X5 2 4 10 12 18 20 26
29	/ 3/	X3 2 3 4 5 10 11 12 13 18 19 20 21 26 27 28
30	/ 4/	X4 X5 2 6 10 14 18 22 26
31	/ 3/	X4 2 3 6 7 10 11 14 15 18 19 22 23 26 27 30
32	/ 3/	X5 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30
33	/ 2/	2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 3

(c) Configuration of original network.

NETWORK DERIVED BY GTMERG
TIME ELAPSED = 74 CENTISECONDS

TRUTH TABLE

```
X1 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
X2 = 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
X3 = 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
X4 = 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
X5 = 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
 1 = 1 0 0 0 1 0 1 1 0 1 0 1 1 0 0 0 0 0 0 0 1 0 0 1 1 1 1 1 0 0 0 0
 2 = 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 3 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 4 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 5 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 6 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 7 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 8 = 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 9 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
10 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
11 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
12 = 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
13 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
14 = 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
15 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
16 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
17 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
18 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
19 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
20 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
21 = 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
22 = 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
23 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
24 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
25 = 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
26 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
27 = 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0
28 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
29 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
30 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
31 = 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0
32 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0
33 = 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
```

(d) Truth table for transduced network

GATE .. LEVEL		FED BY
1	/ 1/	12 21 31 32 33
2	/ 3/	X1 X2 X4 X5
3	/ 1/	
4	/ 1/	
5	/ 1/	
6	/ 1/	
7	/ 1/	
8	/ 3/	X1 X2
9	/ 1/	
10	/ 1/	
11	/ 1/	
12	/ 2/	X1 X3 X5 2
13	/ 1/	
14	/ 3/	X1 X4 X5
15	/ 1/	
16	/ 1/	
17	/ 1/	
18	/ 1/	
19	/ 1/	
20	/ 1/	
21	/ 2/	X2 X3 2
22	/ 3/	X2 X4 X5
23	/ 1/	
24	/ 1/	
25	/ 3/	X2
26	/ 1/	
27	/ 3/	X3
28	/ 1/	
29	/ 1/	
30	/ 1/	
31	/ 2/	X4 14 22 27
32	/ 2/	X5 8 14 22 27
33	/ 2/	14 25 27

* A NETWORK DERIVED BY GTMERG
COST = 12038

(e) Configuration of transduced network.

3. A NETWORK TRANSDUCTION PROCEDURE BY SUBSTITUTING FOR GATES

This section will discuss the NOR-network transduction procedure realized by the FORTRAN program designated NETTRA-G4. This procedure examines every output connection of a selected gate to see if it can be substituted for by connections from external variables and/or existing gates. The concept of compatible sets of permissible functions and a possibly connectable condition† are used in this procedure in searching the candidates of substitution. The substitutions will be performed only when all the output connections of the selected gate can be replaced. Therefore, any possible transduction performed by this procedure will reduce the number of gates in the network.

The input to this program is a description of a particular NOR-network under consideration. This description (explained in great detail in Section 5) consists of a set of various network parameters. The output of this program is the description of the "transduced" network (if a transduction, in this case the substitution of gates, was possible).

The entire NETTRA-G4 program requires 146K bytes of core storage, about 56K bytes being occupied by the actual program instructions and about 90K bytes by the stored data.

The gate substitution procedure is realized by the FORTRAN subroutine PROCV. This subroutine along with the following support subroutines, written in FORTRAN IV for the IBM 360/75, constitutes the program NETTRA-G4: MAIN, PROCV, RQRNW, OUTPUT and SUBNET. Two system-supplied timing subroutines STIMEZ and KTIMEZ are also assumed to be available, but if they are not, their use can be omitted from the program, or

---

†The reader is assumed to be familiar with the definitions presented in [1] and [3].

another suitable timing routine substituted, without changing the procedure itself. The functions of the support subroutines MAIN, OUTPUT, SUBNET will be discussed in Section 4.

The general organization of the program NETTRA-G4 is shown in Fig. 3.1. An arrow from block i to block j represents the fact that the subroutine represented by block i calls the subroutine represented by block j.

Section 3.1 will discuss the procedure itself, and Section 3.2 will show examples obtained by NETTRA-G4.

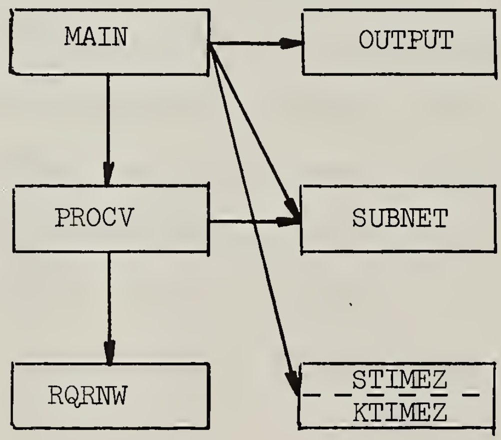


Fig. 3.1 General organization of the program NETTRA-G4.

3.1 Gate Substitution Procedure and Flowchart

The gate substitution procedure is essentially performed by the subroutine PROCV. Subroutine RQRNW, which is called by subroutine PROCV, calculates a compatible set of permissible functions for every output connection of a particular selected gate and for every gate in the network except that particular gate. For the sake of efficiency, this subroutine uses an ordering which is especially designed to fit the purpose of substituting for the selected gate. For this reason, the general subroutine for calculating CSPF's, FORTRAN subroutine MINI2 used in program NETTRA-G3 and some other NETTRA programs, is not used in NETTRA-G4.

Subroutine PROCV is programmed based on the procedure SOGPC described in [1]. The following discussions of PROCV will assume knowledge of the information contained in [1].

The purposes of the main variables and arrays appearing in the subroutine will be explained in the program listing in the Appendex.

PROCEDURE FOR SUBSTITUTION OF GATES

Step 0. Initialize

Step 1. Selection of Gates

According to the ascending order of gate labels, select a gate GI satisfying the following conditions:

1. GI is not an output gate of the network,

2. GI has no single output gate as its immediate predecessor.

If all gates have been selected, return to the calling subroutine (MAIN in NETTRA-G4).

Step 2. Remove Internal Inverters

If GI is the only immediate predecessor of a gate GJ not being an output gate, then GJ is redundant. Remove GJ from the network, and connect all immediate predecessors of gate GI to all immediate successor of gate GJ. Call subroutine SUBNET to update the information on the configuration of the network (successor lists, predecessor lists, etc.), and then go to Step 1.

Step 3. Calculation of CSPF's

Call subroutine RQRNW to calculate CSPF's for all output connections of gate GI, and for all gates except gate GI.

Step 4. Selection of Connections

Select one output connection of gate GI. Let the connection be GIGJ (indicating the connection from gate GI to gate GJ). The connection is selected according to the ascending order of GJ.

If all connections from GI have been considered, go to Step 8.

Step 5. Partition of Candidates for Substitution

Partition external variables and/or existing gates which are not successors of gate GI into the following subsets.

(1) Effectively-connectable functions which satisfy the following conditions

a. must have 0-components corresponding to all 0-components in the CSPF of GIGJ.

b. must have at least one 1-component corresponding to a 1-component in the CSPF of GIGJ.

(2) Possibly-connectable functions which may actually have 1-components corresponding to some 0-components in the CSPF of GIGJ, but the corresponding components in the CSPF's of the functions themselves must be *. The second condition in (1) is also required for a function being in this subset.

(3) All other functions.

Step 6. Selection of Connectable Functions

From connectable functions, select functions which are essential for replacing connection GIGJ.

If these functions contain at least one 1-component corresponding to every 1-component in CSPF of GIGJ, connect these functions to gate GI and call RQRGT (an entry point in subroutine RQRNW) to update CSPF's for all gates feeding these functions, and go to Step 4.

Step 7. Selection of Possibly-Connectable Functions

Select possibly-connectable functions to replace connection GIGJ according to the following substeps.

7-1 Select one possibly-connectable function GK which has at least one 1-component corresponding to a 1-component in the CSPF of GIGJ which is not covered by any other functions having been selected.
If all possibly-connectable functions have been considered, then connection GIGJ is not replaceable, and go to Step 9.

7-2 Find a function GL satisfying that (1) GL is not a successor of GI, (2) connecting GL to GK would make GK a connectable function to GJ without changing the essential 1-components in GK. If no such GL can be found, go to Step 7-1.

7-3 Connect GL to GK and GK to GJ. Call SUBNET to update the information on the network (successor lists etc.), call RQRGT to update the CSPF's for gates feeding GK, and call UPTRTH (another entry point in RQRNW) to update the truth table for gates which are successors of GK.

7-4 If all 1-components in the CSPF of GIGJ are covered by the selected functions, go to step 4; otherwise go to Step 7-1.

Step 8. Removing Gate

Disconnect all connections going to or coming out from gate GI. Call SUBNET to update the information on the network. Go to Step 1.

Step 9. Restoring the Network

Since GIGJ could not be replaced, restore the old network (the network before gate GJ was taken into consideration). Go back to Step 1.

According to the procedure described above, the gates which could not be replaced when they were chosen would not be considered again although they may become replacable after some other gate has been removed. The user can easily change the subroutine MAIN to repeatedly apply this procedure until no further improvement could be found. Another way to do this is simply to change the statement 'go to Step 1' in Step 8 to ' go to Step

The flowchart of subroutine PROCV is shown in Fig. 3.2.

The calculation of CSPF's for gates and connections from the selected gate is accomplished by subroutine RQRNW which is called in

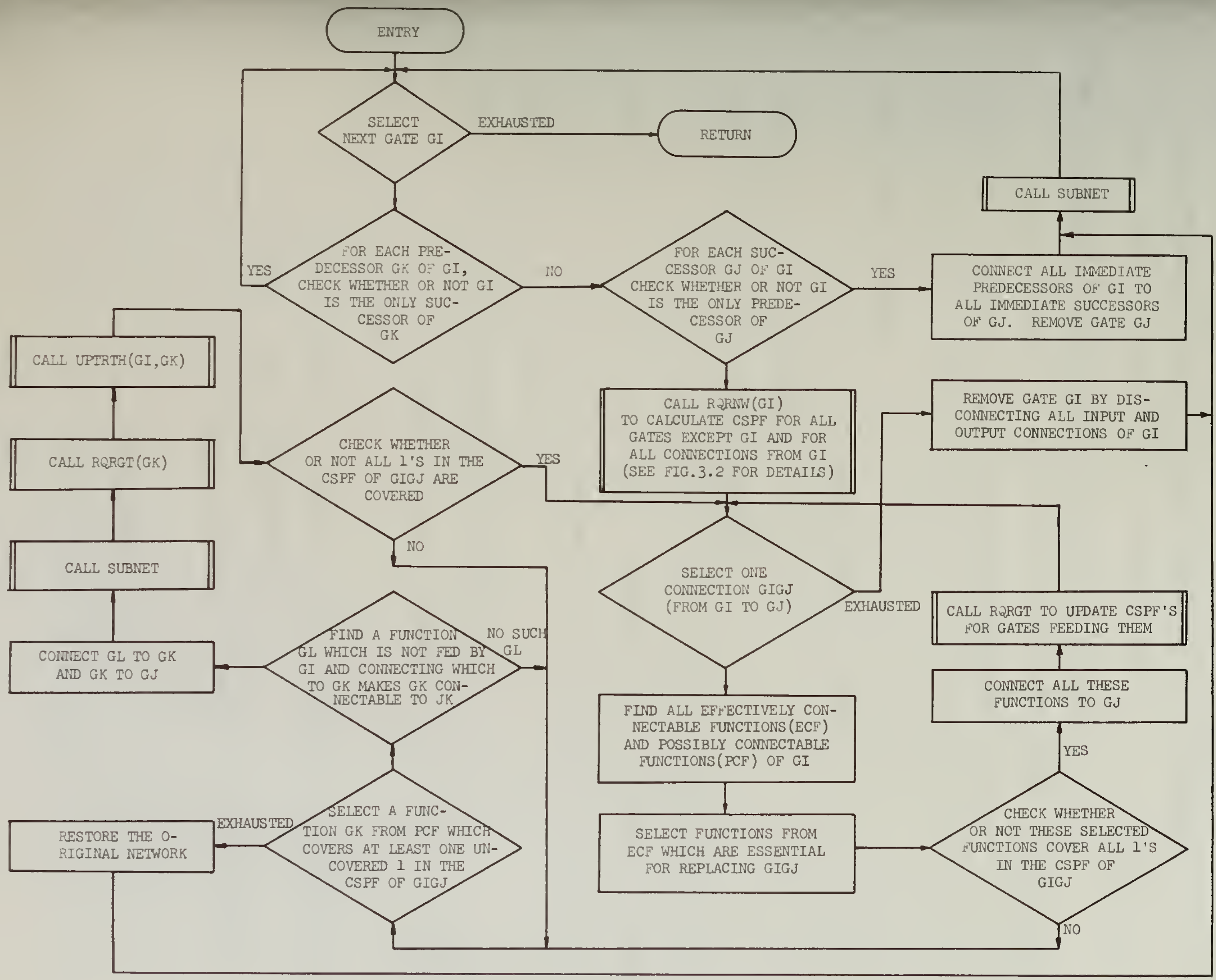


Fig. 3.2 Flowchart of subroutine PROCV.

Step 3 of the above procedure. The CSPF for a gate is calculated by forming intersections of CSPF's for all its output connections. Since the CSPF's for connections (except the connections from the selected gate) need not be stored in the program, only the CSPF for each gate is stored. For the details of the process of the calculation of CSPF's, see [1]. The following is the procedure realized by subroutine RQRNW.

Note that gate GI is specified by the calling subroutine (PROCV) when RQRNW is called.

PROCEDURE FOR CALCULATION OF CSPF's

Step 0. Initialization

Assign * to every component of the CSPF for every gate in the network.

Step 1. CSPF for Output Gates

Assign CSPF's of the output gates in the network the corresponding output functions specified by the input data.

Step 2. Selection of Gates

Select a gate GJ other than GI according to the ordering of gate levels, i.e., gates in a lower level are chosen prior to gates in a higher level, and gates in the same level are chosen according to the ascending order of gate labels.

If all gates have been considered, return to the calling subroutine (PROCV in NETTRA-G4.)

Step 3. Selection of Input Connection

Select an input connection GKGJ of gate GJ, (GKGJ denotes the connection from gate GK to gate GJ). The selection is based on the ascending order of GK with an exception that GIGJ (if exists) is selected last among all input connections to GJ.

Step 4. Calculations of CSPF's for Connections

Calculate the CSPF for input connection GKGJ of gate GJ according to the following rules:

(1) For 1-components in the CSPF of GJ, the corresponding components of CSPF for GKGJ are assigned 0.

(2) For 0-components in the CSPF of GJ, there are the following three cases.

(a) Each of the corresponding components in the CSPF of GKGJ is assigned *, if the corresponding component of the function

of GK is 0.

(b) Each of the corresponding components in the CSPF of GKGJ is assigned 1, if the corresponding component of the function of GK is 1, and no corresponding components of other input connections have been assigned 1 before GKGJ is taken into consideration.

(c) All other components in the CSPF of GKGJ are assigned *'s.

(d) For *-components in the CSPF of GJ, the corresponding components of CSPF of the CSPF of GKGJ are assigned *'s.

Step 5. Calculation of CSPF for Gate†

If GK ≠ GI, take intersection of the CSPF of GKGJ with the intermediate CSPF obtained so far for gate GJ, (i.e., the intersection of CSPF's for input connections of GJ selected prior to GKGJ). This intersection is the new intermediate CSPF for gate GJ (the final CSPF for GJ if GKGJ is selected the last among all input connections to GJ).

Go to Step 3.

The flowchart of this subroutine is shown in Fig. 3.3.

When RQRNW is entered through the entry point RQRGT(GZ), the procedure is applied only to predecessors of gate GZ. When RQRNW is entered through the entry point UPTRTH(GI,GZ), the procedure is applied only to gates which are successors of gate GZ but not gate GI.

---

†This step is actually performed while each component of CSPF for GKGJ is assigned.

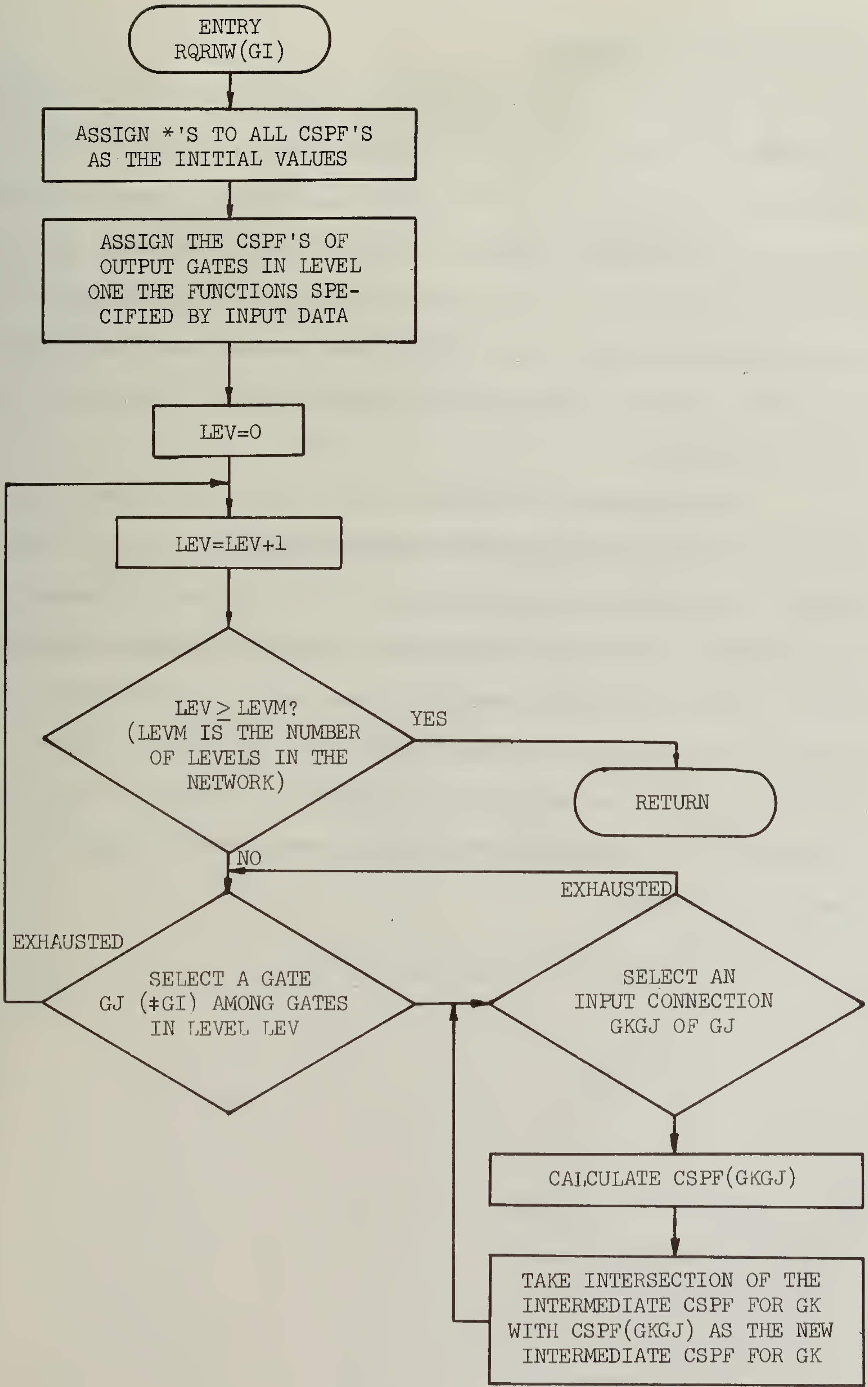


Fig. 3.3 Flowchart of subroutine RQRNW.

3.2 Examples for NETTRA-G4

The printout obtained during the solution of a typical program by NETTRA-G4 is shown in Fig. 3.5. The printout is similar to the one obtained by NETTRA-G3 discussed in Section 2.2 with (a) showing the heading information and parameters for that problem, (b) the truth table of the original network, (c) the configuration of the original network obtained from input cards, (d) the truth table of the transduced network obtained by applying subroutine PROCV, and (e) the configuration of that transduced network.

The procedure for merging of gates and the procedure for substitution of gates can be used in a combined program. Fig. 3.6 shows an example obtained by a combined program which is similar to program NETTRA-G3 with an additional statement 'CALL PROCV' inserted immediately after the statement 'CALL GTMERG' in subroutine MAIN of NETTRA-G3. The original network is the one shown in Fig. 2.3. It can be seen that PROCV improved the network obtained by NETTRA-G3 by reducing 1 gate and 4 connections. However, this does not mean that PROCV is always more powerful than GTMERG.

```
**** 5 VAR. EXAMPLE                  HEX=3B5809F0

     NUMBER OF VARIABLES =    5

     NUMBER OF FUNCTIONS =    1

     COST COEFFICIENT A  = 1000

                     B   =    1

     --- UNCOMPLEMENTED VARIABLES X ---

ORIGINAL NETWORK    COST= 21086
```

(a) Heading information and network parameters.

Fig. 3.5 Printout obtained from NETTRA-G4 for a sample problem.

TRUTH TABLE

X1 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
X2 =	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
X3 =	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
X4 =	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
X5 =	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
1 =	1	0	0	0	1	0	1	1	0	1	0	1	1	0	0	0	0	0	0	0	1	0	0	1	1	1	1	1	0	0	0	0
2 =	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3 =	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4 =	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5 =	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6 =	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7 =	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8 =	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9 =	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11 =	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12 =	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
13 =	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
14 =	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
15 =	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
16 =	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17 =	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0
18 =	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
19 =	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
20 =	1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21 =	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1

(b) Truth table for original network.

GATE .. LEVEL		FED BY
1	/ 1/	3 4 5 7 8 9 10 11 12 13 15 17 21
2	/ 3/	X1 X2 X3 X4 X5
3	/ 2/	X1 X2 X3 X4 2
4	/ 2/	X1 X2 X3 X5 2
5	/ 2/	X1 X2 X3 2
6	/ 3/	X1 X2 X4 X5
7	/ 2/	X1 X2 X4 6
8	/ 2/	X1 X3 X4 X5 2
9	/ 2/	X1 X3 X5 2
10	/ 2/	X2 X3 X4 X5 2
11	/ 2/	X2 X3 X4 2
12	/ 2/	X2 X3 X5 2
13	/ 2/	X2 X3 2
14	/ 3/	X2 X4 X5
15	/ 2/	X2 X4 14
16	/ 3/	X1 X2 X5
17	/ 2/	X2 X5 14 16
18	/ 3/	X2
19	/ 3/	X3
20	/ 3/	X1 X4 X5
21	/ 2/	18 19 20

(c) Configuration of original network.

NETWORK DERIVED BY PROCV
TIME ELAPSED =   142 CENTISECONDS

TRUTH TABLE

```
X1 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
X2 = 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
X3 = 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
X4 = 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
X5 = 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
 1 = 1 0 0 0 1 0 1 1 0 1 0 1 1 0 0 0 0 0 0 0 1 0 0 1 1 1 1 1 0 0 0 0
 2 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 3 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 4 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 5 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 6 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 7 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 8 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 9 = 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
10 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
11 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
12 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
13 = 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
14 = 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
15 = 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0
16 = 1 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
17 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0
18 = 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
19 = 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0
20 = 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
21 = 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
```

(d) Truth table for transduced network.

GATE ..	LEVEL	FED BY
1	/ 1/	9 10 13 15 17 21
2	/ 1/	
3	/ 1/	
4	/ 1/	
5	/ 1/	
6	/ 1/	
7	/ 1/	
8	/ 1/	
9	/ 2/	X1 X3 X5 14
10	/ 2/	X2 X3 X4 X5 16
11	/ 1/	
12	/ 1/	
13	/ 2/	X2 X3 14
14	/ 3/	X2 X4 X5
15	/ 2/	X2 X4 14
16	/ 3/	X1 X2 X5
17	/ 2/	X2 X5 14 15
18	/ 3/	X2
19	/ 3/	X3
20	/ 3/	X1 X4 X5
21	/ 2/	18 19 20

* A NETWORK DERIVED BY PROCV
COST= 1239.

(e) Configuration of transduced network

NETWORK DERIVED BY GTMERG AND PROCV
TIME ELAPSED =   160 CENTISECONDS

TRUTH TABLE

```
X1 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
X2 = 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
X3 = 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
X4 = 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
X5 = 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
 1 = 1 0 0 0 1 0 1 1 0 1 0 1 1 0 0 0 0 0 0 0 1 0 0 1 1 1 1 1 0 0 0 0
 2 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 3 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 4 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 5 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 6 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 7 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 8 = 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 9 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
10 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
11 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
12 = 0 0 1 0 0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
13 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
14 = 1 0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
15 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
16 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
17 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
18 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
19 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
20 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
21 = 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0
22 = 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0
23 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
24 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
25 = 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0
26 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
27 = 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1 0 0 0 0
28 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
29 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
30 = 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
31 = 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 0 1 1 0 0
32 = 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 1 0
33 = 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1
```

(a) Truth table of transduced network

Fig. 3. Printout obtained from a program applying GTMERG and PROCV for the same problem as that shown in Fig. 2.3.

GATE ..	LEVEL	FED BY
1	/ 1/	12 21 31 32 33
2	/ 1/	
3	/ 1/	
4	/ 1/	
5	/ 1/	
6	/ 1/	
7	/ 1/	
8	/ 3/	X1 X2
9	/ 1/	
10	/ 1/	
11	/ 1/	
12	/ 2/	X1 X3 X5 22
13	/ 1/	
14	/ 3/	X1 X4 X5
15	/ 1/	
16	/ 1/	
17	/ 1/	
18	/ 1/	
19	/ 1/	
20	/ 1/	
21	/ 2/	X2 X3 14
22	/ 3/	X2 X4 X5
23	/ 1/	
24	/ 1/	
25	/ 3/	X2
26	/ 1/	
27	/ 3/	X3
28	/ 1/	
29	/ 1/	
30	/ 1/	
31	/ 2/	X4 14 22 27
32	/ 2/	X5 8 14 22 27
33	/ 2/	14 25 27

* A NETWORK DERIVED BY GTMERG AND PROCV
COST= 11034

(b) Configuration of transduced network by GTMERG and PROCV.

4. MAJOR FUNCTIONS OF COMMON SUBROUTINES

The subroutines realizing the procedures presented in the two previous sections share the support of three subroutines whose main functions will be discussed in this section: MAIN, SUBNET, OUTPUT. In addition, subroutine MINI2 described in [2] will also be briefly explained which is used in NETTRA-G3.

Complete program listings of these four subroutines can be found in the Appendix along with the listings of the subroutines realizing the previously described procedures.

The functions of these subroutines are as follows:

MAIN: This subroutine repeatedly reads in groups of input data which include information about the given networks; e.g., the number of external variables, whether or not the compliments of variables are also available as input variables, the number of output functions, the number of NOR gates, the list of connections, and the truth table of the output functions (see Section 5 for details). Using this information, MAIN constructs the incidence matrix, INC$MX, for the network. INC$MX is a two-dimensional array whose arguments represent gates or external variables. An array element INC$MX(A1,A2) $\geq$ 1 indicates a connection from A1 to A2; an array element INC$MX(A1,A2) $\leq$ 0 indicates the absence of a connection from A1 to A2. Next subroutine SUBNET is called to calculate the level of each gate and to make lists of predecessors and successors (i.e., which gates precede which and which gates succeed which). MAIN then prints out the truth table and the constructed incidence matrix of the original network by calling the subroutine OUTPUT. Finally the desired transduction

procedure is applied to the network by calling the subroutine(s) realizing that transduction. The transduced network is stored in INC$MX, replacing the original network. Then MAIN prints the results of the transduction procedure, i.e., lists of removed and added connections, the new incidence matrix and the new truth table.

<u>SUBNET</u>: This subroutine may be entered at three different points by a call to either SUBNET, UNNECE, or PVALUE.

SUBNET generates detailed information on the configuration of the network stored in INC$MX: (1) It calculates the level of each gate in the network. Level 1 is assigned to gates having no output connections thus all gates which have been removed from the network will be assigned level 1. (2) It lists all immediate successors and immediate predecessors for each gate. (3) It calculates the successor matrix which is stored in a two-dimensional array SUC$MX. The value of SUC$MX (A1,A2) indicates the existence or non-existence of a path from gate (or external variable) A1 to gate A2.

UNNECE disconnects certain types of obviously unnecessary connections in the network and updates the above information (discussed in (1), (2), and (3)). The connections removed from the given network are those existing in no paths from the external variables to the output gates.

PVALUE calculates the actual truth table for the entire network stored in INC$MX.

<u>OUTPUT</u>: This subroutine may be entered at five different points by a call to either OUTPUT, PAGE, LINE, TRUTH, OR CKT.

OUTPUT assigns mnemonic names to external variables and gates for the purpose of achieving a readable print-out.

PAGE ejects one page on the printer.

LINE skips a specified number of lines on the print-out sheet. The number is specified by the argument in the call (e.g. "CALL LINE (5)" skips 5 lines).

TRUTH prints out the truth table of the network currently stored in INC$MX.

CKT prints out the network itself.

MINI2: This subroutine is called by GTMERG in NETTRA-G3 for the purpose of calculating CSPF's for gates in a network. Subroutine MINI2 also has the ability to remove some redundant connections. It is described in some detail in [2].

5. INPUT DATA SETUP

In order to fully understand the description of the setup of the input data cards, certain preliminary explanations are necessary.

The purpose of network transductions is to reduce the cost of a network which realizes a certain function (or functions) or to alter the network in such a way as to allow another transduction to eventually accomplish such a reduction. This cost, C, is formally defined by the weighted sum of the number of gates, R, and the number of connections†, I, of a particular network, i.e.,

$$C = A \times R + B \times I$$

where weights A and B are arbitrary non-negative numbers.

Suppose the original network which is to be transformed produces m output functions of n variables. Let x_ℓ, $\ell = 1, \ldots, n$, be the external variables and f_h, $h = 1, \ldots, m$, be the output functions. Before a transformation can be performed on a network by a program, a description of that network must be input to the program. In the case when all of the output functions are completely specified (i.e., no "don't cares"), specifying only the interconnection pattern of the network is sufficient. But if one or more of the output functions is not completely specified, then the user must also provide the truth table (truth tables for all output functions are condensed into a single table) of the problem. Providing the truth table to the program consists of two steps,

---

† A "connection" refers to either a connection from an external variable or an interconnection between two gates.

namely the specification of external variables, and the specification of output functions.

The method of specifying the output functions depends directly upon the method chosen to specify the external variables. External variables may be specified in either of two ways, (a) an implicit specification of external variables, or (b) an explicit specification of external variables.

(a) In the case of implicit specification of external variables, the user specifies the number n of external variables along with a parameter which indicates whether or not the uncomplemented variables are available. Reading the value n along with the parameter, the program internally generates the entries of external variables of an ordinary truth table, that is, a truth table which consists of 2^n input vectors, as shown in Fig. 5.1. In this truth table, the input vectors are arranged according to the order such that an integer j expressed in a binary representation $(x_1 \ldots x_n)$ increases, where x_1 is the most significant digit and x_n is the least significant digit. For example, the truth table for a function of three variables is shown in Fig. 5.2.

The implicit specification of external variables is used for logical design problems in which the output functions have relatively few don't-care terms.

The uncomplemented variables	x_1	$x_1^0 \ldots x_1^j \ldots x_1^{2^n-1}$	
	$\vdots$	$\vdots$	
	x_n	$x_n^0 \ldots x_n^j \ldots x_n^{2^n-1}$	
The complemented variables	$\bar{x}_1$	$\bar{x}_1^0 \ldots \bar{x}_1^j \ldots \bar{x}_1^{2^n-1}$	These entries exist only in the case of logical design problems where the complemented variables are available as external inputs.
	$\vdots$	$\vdots$	
	$\bar{x}_n$	$\bar{x}_n^0 \ldots \bar{x}_n^j \ldots \bar{x}_1^{2^n-1}$	
The output functions	f_1	$f_1^0 \ldots f_1^j \ldots f_1^{2^n-1}$	
	f_2		
	$\vdots$	$\vdots$	
	f_m	$f_m^0 \ldots f_m^j \ldots f_m^{2^n-1}$	

Fig. 5.1 The truth table of output functions of n variables

x_1	0	0	0	0	1	1	1	1	
x_2	0	0	1	1	0	0	1	1	
x_3	0	1	0	1	0	1	0	1	
$\bar{x}_1$	1	1	1	1	0	0	0	0	These entries exist only in the case of logical design problems where complemented variables are available as input variables.
$\bar{x}_2$	1	1	0	0	1	1	0	0	
$\bar{x}_3$	1	0	1	0	1	0	1	0	
f_1	f_1^0	.	.	.	.	.	.	f_1^7	

Fig. 5.2 The truth table of a function of three variables.

(b) In the case of explicit specification of external variables, the user specifies the entries of external variables of the truth table using additional cards called < external-variable-card > s. The explicit specification of external variables is used in the case of logical design problems where output functions have many don't-care terms. Suppose that the output functions are defined for a subset of input vectors of the entire truth table of Fig. 5.1. Let $\vec{x}^{\,j}$, $j = j_1, j_2, \ldots, j_\mu$ denote these input vectors. The user can 'condense' the truth table of Fig. 5.1 into another table shown in Fig. 5.3.

		only μ input vectors				
The uncomplemented variables	x_1	$x_1^{j_1}$	$x_1^{j_2}$	. . .	$x_1^{j_\mu}$	
	.	.				
	.	.				
	.	.				
	x_n	$x_n^{j_1}$	$x_n^{j_2}$	. . .	$x_n^{j_\mu}$	
The complemented variables	$\bar{x}_1$	$\bar{x}_1^{j_1}$	$\bar{x}_1^{j_2}$	. . .	$\bar{x}_1^{j_\mu}$	These entries exist only in the case of logical design problems where the complemented variabl are avaiable as external inputs.
	.	.				
	.	.				
	.	.				
	$\bar{x}_n$	$\bar{x}_n^{j_1}$	$\bar{x}_n^{j_2}$	. . .	$\bar{x}_n^{j_\mu}$	
	f_1	$f_1^{j_1}$	$f_1^{j_2}$	. . .	$f_1^{j_\mu}$	
	.	.				
	.	.				
	.	.				
	f_m	$f_m^{j_1}$	$f_m^{j_2}$	. . .	$f_m^{j_\mu}$	

Fig. 5.3 A 'condensed' truth table having only the input vectors $\vec{x}^{\,j}$, $j = j_1, \ldots, j_\mu$, for which the output functions are defined.

Using < external-variable-card > s, the user can set up internally the table of Fig. 5.3 in place of Fig. 5.1.

5.1 Input Data Card Format

For each separate problem, a set of input data cards must be submitted which consists of the following[†]:

(i) < heading-card >

(ii) < problem-parameter-card >

(iii) < external-variable-card > s

(iv) < output-function-card > s

omitted for certain cases

(v) < connection-description-card > s

Both (i) and (ii) will always consist of only a single card, but (iii), (iv), and (v) may each consist of several cards. Furthermore, types (iii) and (iv) are omitted if all output functions are completely specified, and (iii) need only be prepared in the case of the explicit specification of external variables for the truth table. Following is a description of the formats for each type of input card, (i), (ii), (iii), (iv) and (v):

(i) < Heading-card >

This is the first card of the input deck for a problem. This card may contain any alphanumeric information, in columns 1 ~ 80, which may be used for the identification of the problem, but none of the information on this card will be used in the actual computation. This information will be printed on the first page of the output.

---

† The current implementations of the NETTRA programs accept only heading, problem-parameter, and connection-description cards. Eventually it is hoped that these programs will be modified to accept all of the options described in this section.

(ii) < Problem-parameter-card >

This card specifies the nature of the problem the user wants to solve. There are 7 fields in which to specify the parameters with characters and numerals. These fields are as follows:

Cols. 1 ~ 4: An integer, N, which is right-justified.

This number, N, represents the number of external variables, n, of the output functions. Be sure to punch n (rather than 2n) for N in the case of both complemented and uncomplemented variables available.

Cols. 5 ~ 8: An integer, M, which is right-justified.

This number, M, is the number of output functions, m, to be realized simultaneously. Therefore, of course, M will also be the number of output gates in the network.

Cols. 9 ~ 12: An integer, R, which is right-justified.

This number, R, specifies the number of gates which are included in the network. For various reasons, the user may wish to input networks in which one or more of the gates are "isolated" (i.e., are not connected to any other gates). This is permissible as long as these "isolated" gates are also included in the total number of gates, R.

Cols. 13 ~ 16: An integer, A , which is right-justified.

The number A is the value of the non-negative weight for the number of gates in the cost function. (See Table 5.1.1, 'Typical combinations of values A and B for different network reduction problems'.)

Cols. 17 ~ 20: An integer, B , which is right-justified.

The number B is the value of the non-negative weight for the number of connections in the cost function. (See Table 5.1.1.)

Col. 24: A blank 'b'[†], or one of the characters, 'C', 'X', 'Y', 'U' or 'V'.

The 'b' or 'C' parameter represents an implicit specification of both the external variables and an implicit specification of the output functions (in this case, the output functions will be calculated from the connection pattern of the network). The 'X' or 'Y' parameter indicates an implicit specification of external variables only. The 'U' or 'V' parameter indicates an explicit specification of external variables. (See summary of these symbols in Table 5.1.2)

The 'b' or 'X' parameter specifies that only uncomplemented external variables are available for the network. The 'C' or 'Y' parameter specifies that both uncomplemented and complemented variables are available for the network. If the user specifies the 'b', 'X', 'C', or 'Y' parameter, the program sets up the truth table by generating a set of 2^n input vectors $(x_1^j, \ldots, x_n^j)$, for $j = 0, \ldots, 2^n - 1$, in the case of a 'b' or 'X' parameter, or $(x_1^j, \ldots, x_n^j, \bar{x}_1^j, \ldots, \bar{x}_n^j)$ for $j = 0, \ldots, 2^n - 1$, in the case of a 'C' or 'Y' parameter.

The 'b' or 'C' parameters should be used for problems in which the output functions contain no don't-care terms. For such problems, the preparation of the < external-variable-card > s and the < output-function-card > s can be dispensed with since the program can calculate completely all output functions using only a description of the

---

† A 'b' stands for a blank (i.e., no character punched).

Network Reduction Problem	Values of A and B
reducing the number of gates only.	A = 1 and B = 0
reducing the number of gates primarily, then reducing the number of connections secondarily.†	A = 100 and B = 1
reducing the number of connections only.	A = 0 and B = 1
reducing the number of connections primarily, then reducing the number of gates secondarily.	A = 1 and B = 100
reducing the sum of the number of gates and the number of connections.	A = B = 1

Table 5.1.1 Typical combinations of values A and B for different network reduction problems.

† Most of the programs in the NETTRA system are oriented toward this reduction problem, so the user will probably find this combination of A and B the most useful.

uncomplemented variables only available	both complemented and uncomplemented variables available	
'b'	'C'	implicit specification of external variables and output functions
'X'	'Y'	implicit specification of external variables
'U'	'V'	explicit specification of external variables

Table 5.1.2 Possible symbols for column 24 of < problem-parameter-card >.

connection pattern of the network (provided by the <connection-description-card>s).

Similarly, the 'X' or 'Y' parameter implies the use of a complete truth table (i.e., 2^n input vectors for n external variables) inside the program. Since from this information the program can easily generate the truth table entries for the external variables, as just mentioned, the < external-variable-card > s are unnecessary. The m < output-function-card > s, however, must still be prepared.

The 'U' parameter specifies that only uncomplemented external variables are available for the network. The 'V' parameter specifies that both uncomplemented and complemented variables are available for the network. In either case, the 'U' or the 'V' parameter, the user must prepare n < external-variable-card > s and m < output-function-card > s. The program sets up the truth table by reading these < external-variable-card > s and < output-function-card > s.

Cols. 25 ~ 28: An integer, NEPMAX, which is right-justified.

This parameter is omitted for all NETTRA programs except those involving "error-compensation" routines. In the cases where NEPMAX is required, a further discussion of this parameter can be found elsewhere in the manual. The abbreviation NEPMAX is a mnemonic for "maximum number of error positions", and the default is $NEPMAX = 2^{(n - 1)}$, where n is the number of external variables.

(iii) < External-variable-card > s

In combination with the 'U' or 'V' parameter in column 24 of the < problem-parameter-card >, the n < external-variable-card > s specify the entries of external variables of the truth table of

Fig. 5.3. Each card contains the binary representation of external variable x_ℓ, i.e., $(x_\ell^{j1}, x_\ell^{j2}, \ldots, x_\ell^{j\mu})$, starting from column 1 of the card. The maximum number of bits in a binary representation is limited to 32. (This means the maximum number of input vectors is 32.) If the actual number of bits is less than 32, then a termination symbol '/' (slash) is put on the right of the right-most bit of the binary representation on the first < external-variable-card >. The remaining columns after the termination symbol '/' in the first card, as well as the same columns in the following cards, may contain any alphanumeric information which may be used for identification. This information will not be printed on the output pages.

In the case of the 'V' parameter, the program generates the binary representations corresponding to complemented variables by taking negations of the entries of the < external-variable-card > s. Therefore the user <u>must not</u> provide < external-variable-card > s representing the complemented variables, $\bar{x}_\ell$.

If one of the parameters 'b', 'C', 'X', or 'Y' appears in column 24 of the < problem-parameter-card >, the user <u>does not</u> prepare < external- variable-card > s.

(iv) < Output-function-card > s

The m < output-function-card > s specify the set of m output functions to be realized simultaneously. Each card contains the binary representation of one output function f_h, starting from column 1 of the card. A symbol '*' is used to denote don't-care terms, if any. The maximum number of bits in a binary representation is limited to 32.

The actual number of bits must be 2^n in the case of an implicit specification of external variables, or must be the same as defined by the < external-variable-card > s in the case of an explicit specification of external variables. The remaining columns, up to column 72 (inclusive), may contain any alphanumeric information which may be used for identification. This information will not be printed on the output pages.

If either the 'b' or 'C' parameter appears in column 24 of the < problem-parameter-card >, the < output-function-card > s must be omitted.

(v). < Connection-description-card > s

In the present version of the program, 9 cards (some of which may be just blank cards) are required.† Each of these cards is divided into 16 fields of 5 columns each (i.e., columns 1 ~ 5, 6 ~ 10, 11 ~ 15, ..., 71 ~ 75, 76 ~ 80). Beginning with the first field of the first card, continuing through the succeeding fields of that card and through the fields of as many additional cards as necessary (up to a maximum of 9, total), the expressions (explained in the next paragraph) C_1, C_2, C_3, ..., are punched right-justified in their respective fields.

Each gate of the network is labeled uniquely by assigning it one of the integers 1, 2, ..., R, such that the output gates receive

---

† For many uses, the user will probably find that these 9 cards far exceed his needs, and may thus be inconvenient. In such a case, the number of required cards may be easily adjusted by making the obvious changes in two statements (A READ statement and a DO statement) following the comment card "C***** READ IN NETWORK INFORMATION AND SET UP INC$MX *****" in subroutine MAIN.

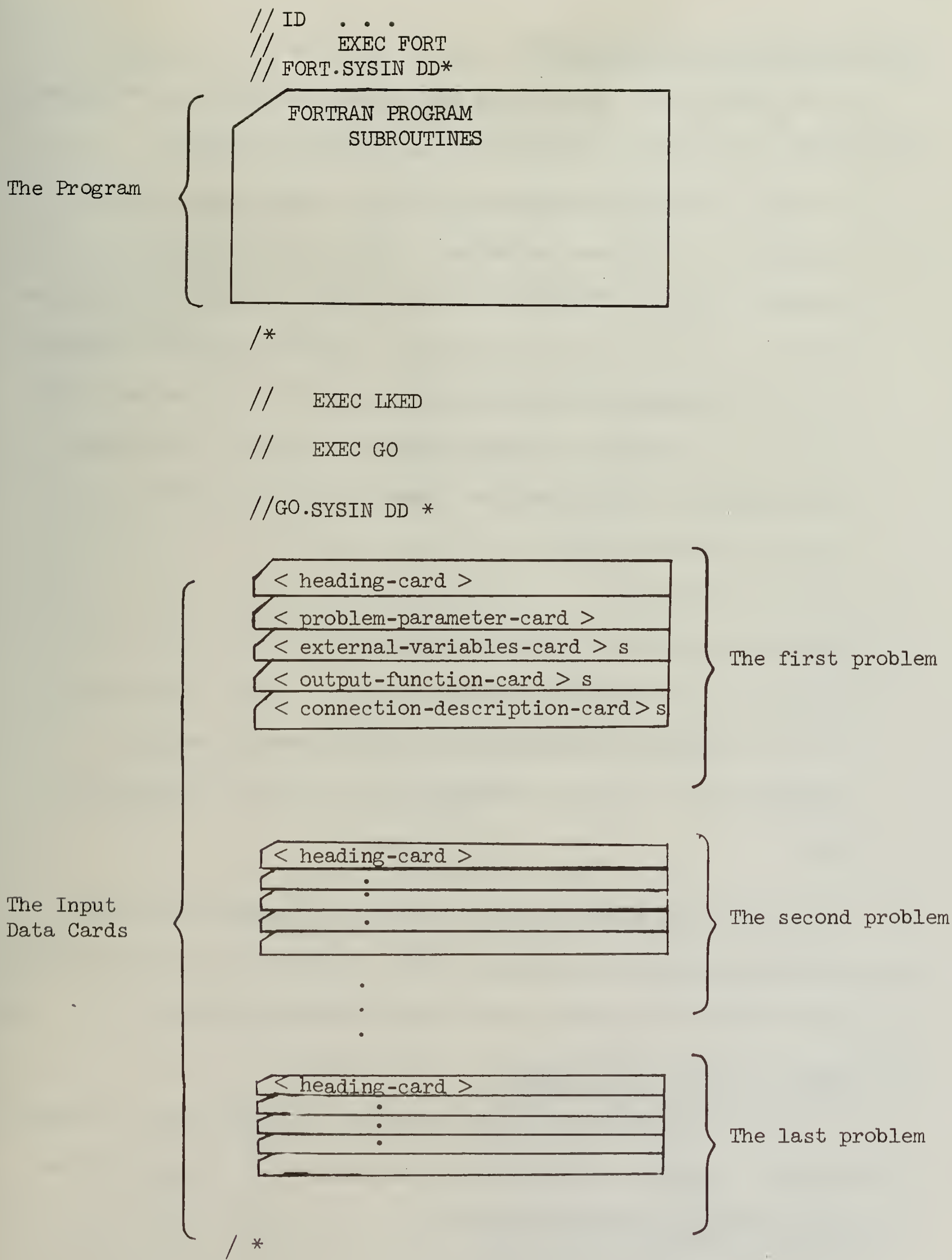
the labels 1, 2, ..., m. The names X1, X2, ..., Xn are assigned to the external variables $x_1, x_2, \ldots, x_n$ (and the names Y1, Y2, ..., Yn to the complemented external variables $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n$, if appropriate).† Now, for each connection of the network (i.e., including <u>both</u> the connections from external variables to gates and connections from gates to other gates), a 4 character expression, C_i, is formed, to represent that connection as follows: to represent a connection from gate GI to gate GJ, the numeric label GI is inserted into the first two character positions of C_i and the numeric label GJ is inserted into the second two positions (e.g., the C_i for a connection from gate 9 to gate 5 would be "0905"); to represent a connection from external variable XI to gate GJ, the alphanumeric label XI is inserted into the first two character positions of C_i and the numeric label GJ into the second two positions (e.g., the C_i for a connection from external variable x_3 to gate 10 would be "X310").

Every connection of the network must be represented by a C_i, although there are no restrictions on the order in which the connections (i.e., C_i's) are punched onto the input cards.

---

† At the time of writing, the programs have not yet been changed to recognize this new labeling system. The old labels are simply: 1, 2, ..., n, for external variables $x_1, x_2, \ldots, x_n$ (and $n+1$, $n+2, \ldots, 2n$ for the complemented variables $\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_n$, if they are permitted in the problem); $n+1, n+2, \ldots, n+m$ for the m output gates of the network ($2n+1, 2n+2, \ldots, 2n+m$ if complemented variables are included); and finally $n+m+1, n+m+2, \ldots, n+R$ ($2n+m+1, 2n+m+2, \ldots, 2n+R$) for the non-output gates of the network.

These five groups of cards, (i), (ii), (iii), (iv) and (v) in the correct order constitute the necessary description for a single problem. In order to solve several problems during the same computer run, the descriptions for the desired problems are just arranged serially. See Fig. 5.1.1 for an example of the sequencing of all cards for the execution of a NETTRA program using typical JCL statements for the IBM 360/75.



<u>Fig. 5.1.1</u> Input card sequence for the execution of a typical NETTRA program.

5.2 Restrictions on Problem Size

In order to fit the programs into a finite amount of space, some restrictions on the size of an acceptable problem are required:

1. The number t of input vectors in the truth table is 32 or less.
2. The number n of external variables.

   Because of $t \leq 32$, n is 5 or less in the case of completely specified functions. In the case of incompletely specified functions, however, any $n \leq 20$ is acceptable if only uncomplemented variables are available, or $n \leq 10$ if both uncomplemented and complemented variables are available, provided that the truth table is defined by the < external-variable-card > s.
3. The number R of gates.

   The number of gates, R, may not exceed 40-n in the case of only uncomplemented variables available (a 'b', 'X', or 'U' parameter). In the case of both uncomplemented and complemented variables available (a 'C', 'Y' or 'V' parameter), the limit is lowered to 40-2n.

All of these limitations are essentially imposed by the array sizes in the programs as presently written. To loosen the restrictions is mainly a task of increasing array dimensions appropriately.

5.3 Examples of Input Data Setup

The following examples will illustrate, for the general program in the NETTRA system, various possible input data card setups complying with the directions given in Section 5.1.

Example 1: A two output network of four variables shown in Fig. 5.3.1. Assume the two output functions are $f_1 = \mathrm{CCEF}$[†] and $f_2 = \mathrm{3BBB}$ and only uncomplemented variables are available. Furthermore, assume it is desired to reduce the number of gates primarily and the number of connections secondarily (see Table 5.1.1).[††]

From this description, the < problem-parameter-card > must contain the following values:

Cols. 1 ~ 4	4,	the number of external variables
Cols. 5 ~ 8	2,	the number of output functions
Cols. 9 ~ 12	8,	the number of gates in the original network
Cols. 13 ~ 16	100,	the value of A
Cols. 17 ~ 20	1,	the value of B
Cols. 24	'b',	uncomplemented variables only available and implicit specification of both the external variables and the output functions
Cols. 25 ~ 28	'b',	since the NEPMAX parameter is unrelated to the program to be used

Fig. 5.3.2 shows the setup of data cards used to specify the network in Fig. 5.3.1 as input for the program. Notice that in forming the C_i, the four uncomplemented variables are represented by the labels X1, X2, X3, X4; the two output gates by the numbers 1, 2; and the remaining gates, by the numbers 3, 4, 5, 6, 7, 8. This manner of labeling is

---

† For convenience, the output functions are expressed in hexidecimal notation. When the numbers in this notation are expanded into binary, they represent the output vectors as they appear (i.e., in the same left-to-right order) in the complete truth table described earlier and pictured in Fig. 5.1.

†† This assumption is implicit in most of the transduction procedures and their implementations which comprise the NETTRA system.

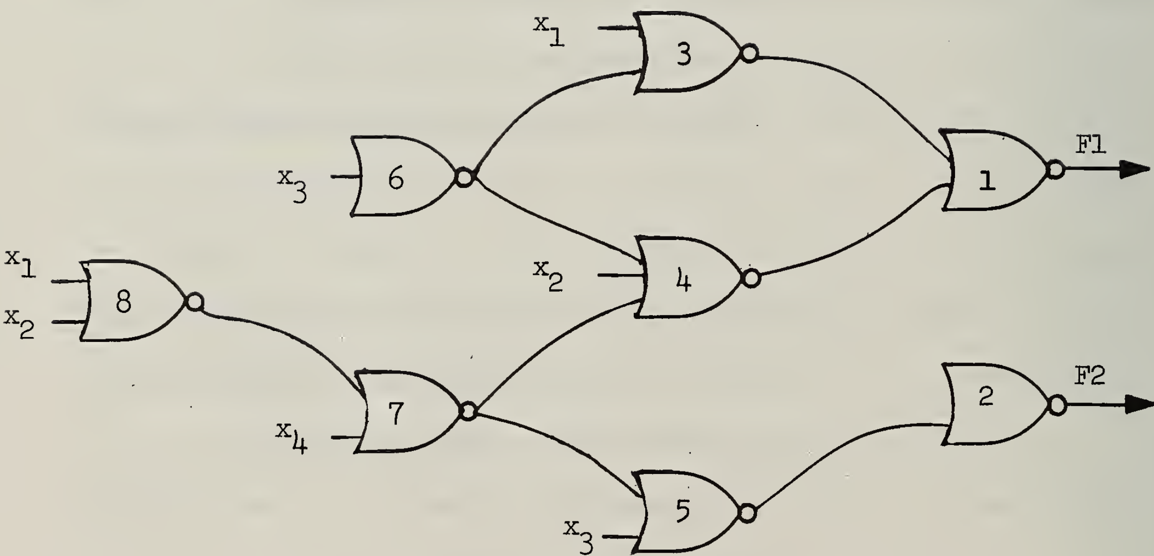


Fig. 5.3.1 Network to be transformed in Examples 1 and 2.

```
Column No.                   00000000011111111112222222222333333333344444...
                             12345678901234567890123456789012345678901234...

8 blank
 connection-                 .
 description-                .
 cards                       .

1st connection-
description-card             __301__401__502_X103__603__604_X204__704__705_X305_X306__807_X407_X108_X208

problem-
parameter-card               ___4___2___8_100___1

heading-card                 _2_OUTPUT,_4_VARIABLE_NETWORK___1ST_FUNCTION_=_CCEF___2ND_FUNCTION_=_3BBB
```

Fig. 5.3.2 Possible setup of data cards to specify the problem given in Example 1.

strictly required by the instructions for preparing the < connection-description-card > s (see Section 5.1).

The heading card in Fig. 5.3.2 will simply be read by the program and printed character for character onto the output page as an identification of the particular problem. Below that, the number of variables, number of functions, and the cost coefficients, A and B, will be printed (all with appropriate labels). Also, immediately following will be a statement of what types of external variables are permitted (i.e., either just uncomplemented variables or both complemented and uncomplemented) along with their generic names:

$$\left.\begin{array}{l} \text{X - for uncomplemented variables} \\ \text{Y - for complemented variables} \end{array}\right\} \begin{array}{l} \text{if external variables} \\ \text{were implicitly specified} \end{array}$$

<u>or</u>

$$\left.\begin{array}{l} \text{U - for uncomplemented variables} \\ \text{V - for complemented variables} \end{array}\right\} \begin{array}{l} \text{if external variables} \\ \text{were explicitly specified} \end{array}$$

For example, if both X and Y appear as generic names (as would occur in the case of an implicit specification of external variables with both complemented and uncomplemented variables available) then the external variable names which appear on subsequent output pages will be X1, X2, ..., Xn and Y1, Y2, ..., Yn. Or, if both U and V appear as generic names (as would occur in the case of an explicit specification of external variables with both complemented and uncomplemented variables available) the external variable names which appear in the output will be U1, U2, ..., Un (for the uncomplemented variables) and V1, V2, ..., Vn (for the complemented variables). It should be noted, however, that the letters U and V, as used as replacements for X and Y (respectively) in the

naming of external variables (e.g. U1, V1 instead of X1, Y1), appear strictly on the output pages of the program - they are not used internally in the program and they must not appear in the variable names punched on the < connection-description-card > s by the user. They are intended only as an aid to the user so that, at a glance at the transformed network in the output, he can easily distinguish whether the external variables were implicitly or explicitly specified for that particular problem.

Following the statement of whether only uncomplemented or both complemented and uncomplemented external variables are employed, the user will find next on the output page the cost of the original network which was input to the program. This is the cost which was defined in the beginning of Section 5.

The cost will be followed by a truth table (generally in the same form as Fig. 5.1) showing the outputs (0 or 1) of all of the gates in the network for every external variable input combination (i.e., combinations of 0's and 1's) of interest.

Finally, below the truth table will be printed a description of the network submitted as input. This is for documentation purposes, and it is also much more readable than the network description which appeared on the < connection-description-card > s. In this description, each gate is listed along with the names of the gates and external variables which feed it. Also, to assist the user in sketching the network from its description, the level of each gate in the network is included (gates which do not feed other gates are assigned to level 1, all other gates are assigned level numbers such that each gate is in a level one

higher than the highest level gate directly fed by it).

All of the information just described will be printed before the execution of the transduction actually begins. This will be followed, beginning at the top of a new output page, by the network(s) actually obtained as a result of the computation. First the complete truth table of the transformed network will be printed, followed by a network connection description of the form just described above. Finally, the cost of the new network will be calculated and printed.

In this example, it was assumed that there were no "don't-cares" in the output functions implicitly specified by the input, thus no < external-variable-card > s or < output-function-card > s were included. In the next example, however, < output-function-card > s will be required in order to specify some of the components of the output functions as "don't-cares".

<u>Example 2</u>: The two output network of four variables shown in Fig. 5.3.1. This is the same network used in Example 1, but this time the output functions are not assumed to be completely specified. Let f_1 = '11001**01*10*111' and f_2 = '0**110111*111011' be the required functions. Also, suppose that both complemented and uncomplemented variables are desired to be available during the transduction. Again the problem is to reduce the number of gates primarily and the number of connections secondarily.

For this problem, the following values must appear on the < problem-parameter-card >:

Cols.	1 ~ 4	4, the number of external variables
Cols.	5 ~ 8	2, the number of output functions

Cols.	9 ~ 12	8,	the number of gates in the original network
Cols.	13 ~ 16	100,	the value of A
Cols.	17 ~ 20	1,	the value of B
Col.	24	Y,	indicative of an implicit specification of external variables and the availability of both complemented and uncomplemented variables

Fig. 5.3.3 shows the setup of the data cards corresponding to this problem. Notice the differences and similarities to the data cards shown in Fig. 5.3.2. The < problem-parameter-card > differs only in column 24. The < external-variable-card > s are missing in both Fig. 5.3.2 and Fig. 5.3.3 since the external variables are implicitly specified for both problems. The < output- function-card > s, however, appear in Fig. 5.3.3 but not in 5.3.2 since they are necessary to specify "don't-care" components which do not occur in the completely specified output functions of Example 1. In both cases, though, the < connection-description-card > s are identical since the original networks are identical.

By allowing "don't-care" terms in the output functions, and by allowing the use of both complemented and uncomplemented variables[†] (even though the original network employed only uncomplemented variables), the restrictions during the transduction process are loosened (compared to what they were for Example 1), perhaps permitting a network of

---

† In the case of NETTRA-PG1, -P1, and -P2, it is useless to specify Y rather than X in column 24 for this example. Since the original network uses only uncomplemented variables, to these programs which perform "pruning" procedures (i.e., procedures which are incapable of adding new connections) the availability of complemented variable is not meaningful.

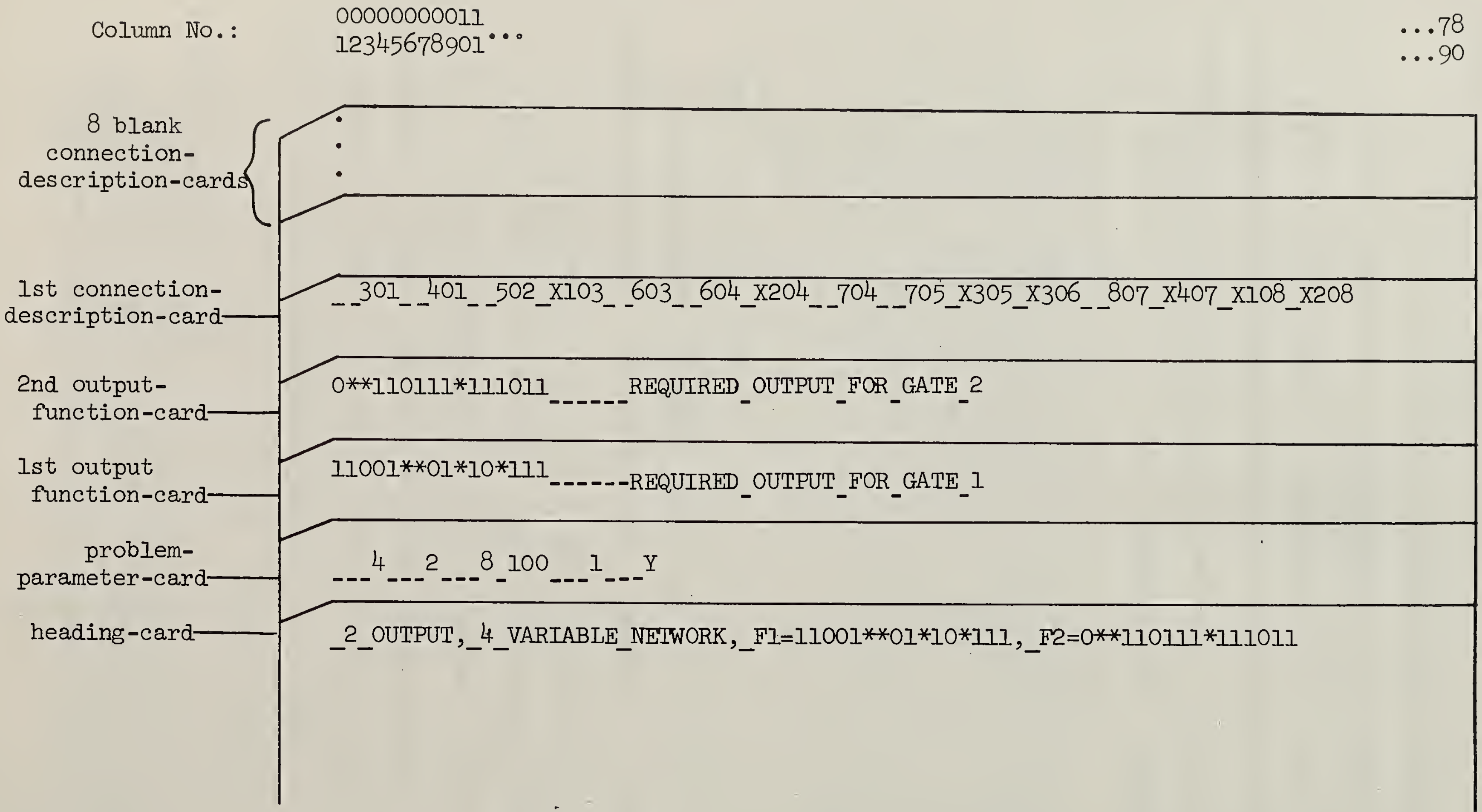


Fig. 5.3.3 Possible setup of data cards to specify the problem given in Example 2.

less cost to be obtained.

Notice that the first < output-function-card > corresponds to the output of gate 1 and the second < output-function-card > corresponds to the output of gate 2. This must hold true for every problem in which < output-function-card > s are included; the gates labeled 1, 2, ..., m must correspond to the output functions specified on < output-function-card > s 1, 2, ..., m, respectively.

Of course, the printed output of the program will be in the same format described in Example 1.

<u>Example 3</u>: The three output network of six variables shown in Fig. 5.3.4. The outputs are again assumed to be incompletely specified. In fact, only the following 11 input combinations are specified out of a possible 64 ($= 2^6$):

x_1	0	0	0	0	0	0	0	0	0	0	1
x_2	0	0	0	0	0	0	0	1	1	1	0
x_3	0	0	0	0	0	0	0	0	0	1	1
x_4	0	0	0	0	1	1	1	0	0	0	1
x_5	0	0	1	1	0	0	1	0	1	1	0
x_6	0	1	0	1	0	1	0	1	1	0	0
F_1	0	0	1	1	0	0	*	0	0	0	0
F_2	1	1	*	1	1	1	0	1	1	0	*
F_3	1	1	0	0	0	0	0	1	0	0	0

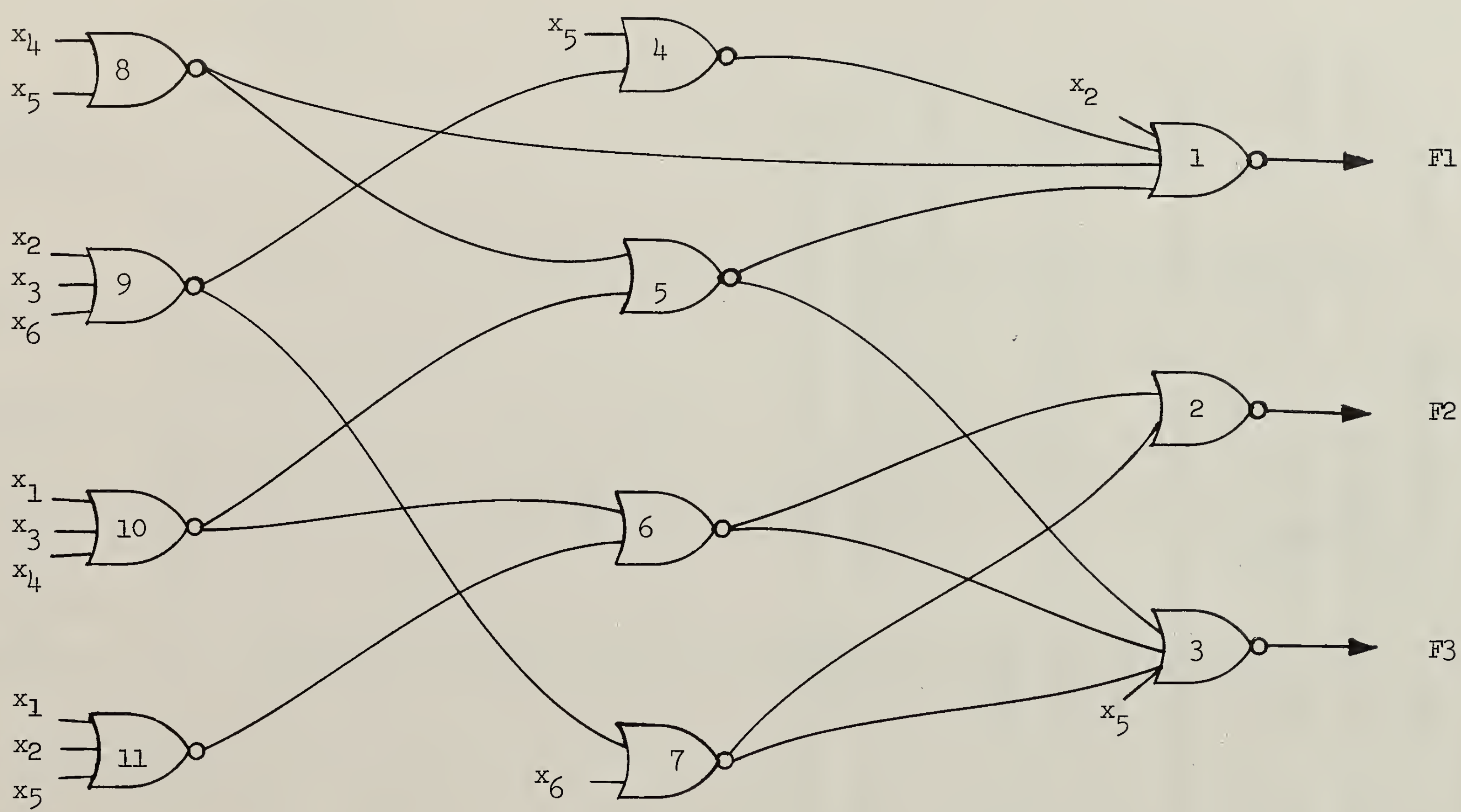


Fig. 5.3.4 Network to be transformed in Example 3.

Additionally, only uncomplemented variables are assumed to be available, and the problem is to reduce the number of gates primarily and the number of connections secondarily.

For this example, the following parameters appear on the < problem-parameter-card >:

Cols.	1 ~ 4	6,	the number of external variables
Cols.	5 ~ 8	3,	the number of output functions
Cols.	9 ~ 12	11,	the number of gates in the original network
Cols.	13 ~ 16	100,	the value of A
Cols.	17 ~ 20	1,	the value of B
Col.	24	U,	indicative of an explicit specification of external variables and the availability of only uncomplemented variables

Fig. 5.3.5 shows a possible setup of the data cards corresponding to this example. Notice that in this example, the <external-variable-card > s are included, whereas in the two previous examples they were omitted. Although this problem is not too realistic (none of the 3 functions is actually a 6-variable function), it demonstrates the input data preparation to be used in cases where many external variables are present and a high percentage of "don't care" terms exist.

Again, the printed output from the program will follow the same format described in Example 1.

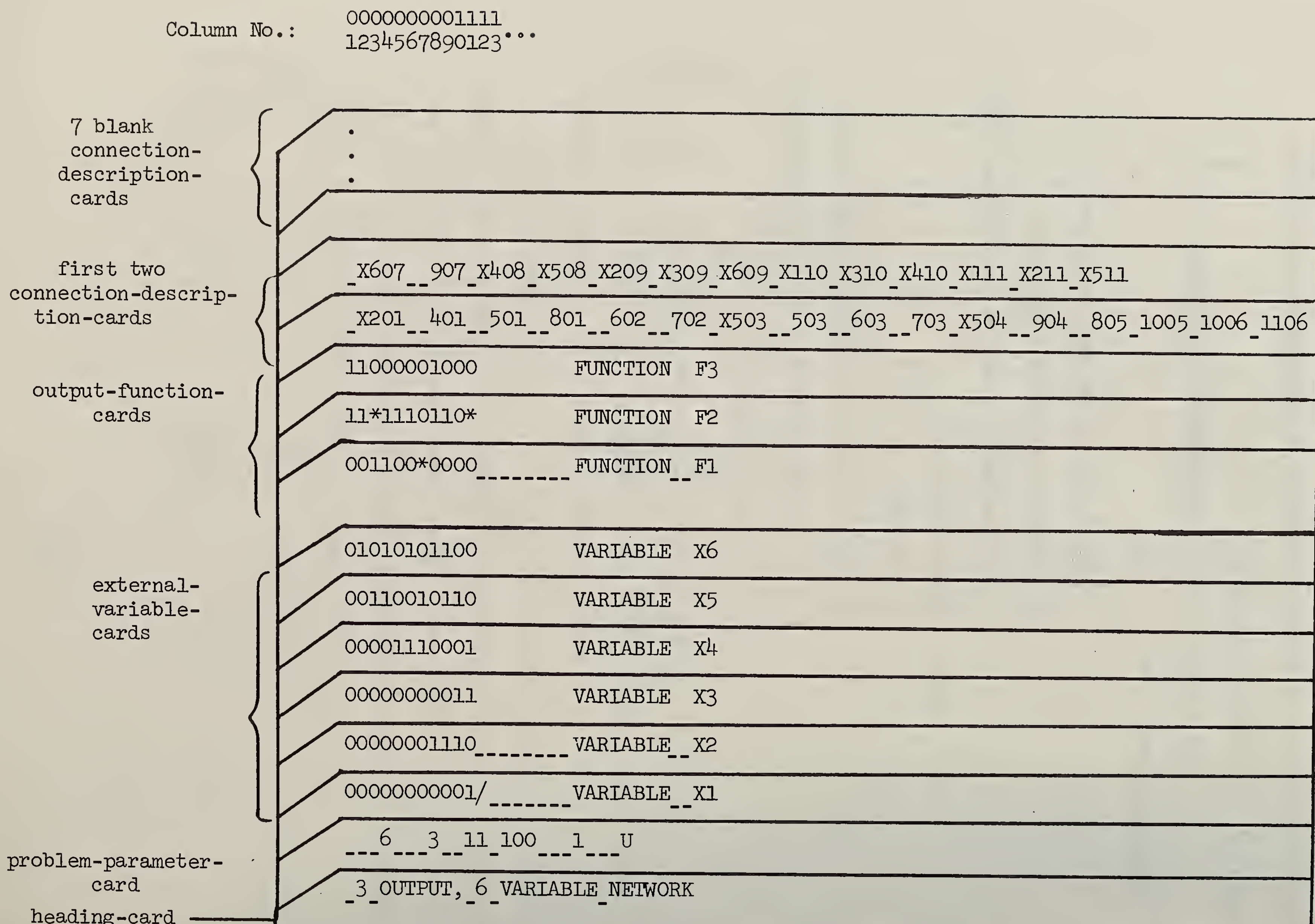


Fig. 5.3.5 Possible setup of data cards to specify the problem given in Example 3.

REFERENCES

[1] H.C. Lai and Y. Kambayashi, "NOR Network Transduction by Generalized Gate Merging and Substitution (Principles of NOR Network Transduction Programs NETTRA-G3 and NETTRA-G4)", to appear as Report, Department of Computer Science, University of Illinois, Urbana, Illinois.

[2] H. C. Lai and J. N. Culliney, "Program Manual: NOR Network Pruning Procedures Using Permissible Functions (Reference Manual of NOR Network Transduction Programs NETTRA-PG1, NETTRA-P1, and NETTRA-P2)", to appear as Report, Department of Computer Science, University of Illinois, Urbana, Illinois.

[3] Y. Kambayashi and S. Muroga, "Network Transduction Based on Permissible Functions (General Principles of NOR Network Transduction NETTRA Programs)", to appear as Report, Department of Computer Science, University of Illinois, Urbana, Illinois.

[4] J. N. Culliney, H. C. Lai, and Y. Kambayashi, "Pruning Procedures for NOR Networks Using Permissible Functions (Principles of NOR Network Transduction Programs NETTRA-PG1, NETTRA-P1, and NETTRA-P2)", to appear as Report, Department of Computer Science, University of Illinois, Urbana, Illinois.

APPENDIX
Program Listings

Following are the FORTRAN program listings for the two programs, NETTRA-G3 and NETTRA-G4, respectively, discussed in this manual. Many of the variables and arrays used are defined in the program themselves.

```
C
C
C***********************************************************************
C                                                                      *
C                                                                      *
C         PPPP    RRRR     OOO     GGG    RRRR      A      M     M     *
C         P   P   R   R   O   O   G   G   R   R    A A     MM   MM     *
C         P   P   R   R   O   O   G       R   R   A   A    M M M M     *
C         PPPP    RRRR    O   O   G  GG   RRRR    AAAAA    M  M  M     *
C         P       R  R    O   O   G   G   R  R    A   A    M     M     *
C         P       R   R    OOO     GGG    R   R   A   A    M     M     *
C                                                                      *
C                                                                      *
C                                                                      *
C N   N   EEEEE   TTTTT   TTTTT   RRRR     A             GGG    33333  *
C NN  N   E         T       T     R   R   A A           G   G      3  *
C N N N   E         T       T     R   R  A   A          G         3   *
C N  NN   EEE       T       T     RRRR   AAAAA  XXXXX   G  GG      3  *
C N   N   E         T       T     R  R   A   A          G   G  3   3  *
C N   N   EEEEE     T       T     R   R  A   A           GGG    333   *
C                                                                      *
C                                                                      *
C***********************************************************************
C
C
      IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)
C     EDITION BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
C
C     NOTE: ALL COMMON VARIBLES MIGHT NOT BE USED IN THIS PROGRAM.
C
C     COMMON VARIABLES:
C        $GT: POINTS TO A 'COLUMN' OF POTAB. FOR EACH 'ROW' THE ENTRY
C             IN THIS COL. TELLS GATE WHERE FN. IS REALIZED.
C       $LTH: POINTS TO A 'COLUMN' OF POTAB.  FOR EACH 'ROW' THE ENTRY
C             IN THIS COL. TELLS HOW MANY CONNECTIONS MUST BE ADDED.
C       $NOE: POINTS TO A 'COLUMN' OF POTAB.  FOR EACH 'ROW' THE ENTRY
C             IN THIS COL. TELLS THE NUMBER OF 1-ERRORS CREATED IF THIS
C             ROW IS USED.
C        $PW: POINTS TO A 'COLUMN' OF POTAB.  FOR EACH 'ROW' THE ENTRY
C             IN THIS COLUMN TELLS THE PREFERENCE WEIGHT.
C          A: WEIGHT FOR NO. OF GATES IN COMPUTING COST FUNCTION.
C          B: WEIGHT FOR NO. OF CONNECTIONS IN COMPUTING COST FUNCTION.
C       COST: COST OF NETWORK - A MEASURE OF NETWORK SIZE.
C     ESS1S: RECORDS NO. OF ESSENTIAL 1'S IN EVERY INPUT TO CURRENT GCO
C             (POSITIONS IN ESS1S CORRES. TO GATES NOT FEEDING GCO ARE
C             IGNORED).
C     F$UB1: POINTS TO LAST ELEMENT IN F$1.
C        F$1: LISTS (CONSECUTIVELY) POSITIONS OF DESIRABLE 1'S (FOR
C             COVERING) IN A CONNECTIBLE FUNCTION.
C         GI: LABEL OF A PARTICULAR GATE.
C     GLEVEL: GLEVEL(GI) TELLS WHICH LEVEL OF THE NETWORK GI IS IN.
C     GSMALL: STORES INTERMEDIATE AND FINAL CALCULATED CSPF'S.
C      HLIST: HLIST(I,J) GIVES NAME OF I-TH GATE (OR EX. VAR.) IN NET-
C             WORK LEVEL J.
C       IDXO: LIST OF O-COORDINATES IN CSPFE OF THE GATE UNDER
C             CONSIDERATION.
C      IDXOE: LIST OF O-ERROR-COORDINATES IN CSPFE OF THE GATE UNDER
C             CONSIDERATION.
C       IDX1: LIST OF 1-COORDINATES IN CSPFE OF THE GATE UNDER
C             CONSIDERATION.
C      IDX1E: LIST OF 1-ERROR-COORDINATES IN CSPFE OF THE GATE UNDER
```

```
C              CONSIDERATION.
C        IFLAG: SAME AS EYEFLG IN SUBROUTINE PROCII.
C      INC$MX: INC$MX(GI,GJ)>0 MEANS THERE EXISTS A CONNECTION FROM GATE
C              (OR EX. VAR.) GI TO GATE GJ.  INC$MX(GI,GJ)=0 IF NOT.
C      INPTCV: LISTS FOR EACH CORRESPONDING ENTRY OF F$1, HOW MANY INPUTS
C              HAVE A '1' IN THE POSITION INDICATED BY F$1.
C       IPATH: IPATH(GI)=1 MEANS GATE GI IS ON A PATH FROM A CERTAIN GATE
C              TO AN OUTPUT GATE.  OTHERWISE IPATH(GI) = 0.
C       IPRED: IPRED(I,GJ) GIVES THE NAME OF THE I-TH GATE OR EX. VAR. IN
C              A LIST OF GATES AND EX. VAR. FEEDING GJ.
C       ISUCC: ISUCC(I,GJ) GIVES THE NAME OF THE I-TH GATE FED BY GJ.
C       JFLAG: SAME AS JAYFLG IN SUBROUTINE PROCII.
C        KEYA: A FLAG INDICATING IF ANY ERROR COMPENSATION HAS BEEN
C              PERFORMED.
C        KEYB: A FLAG INDICATING IF ANY PRIMARY 0-ERROR-COORDINATES HAS
C              BEEN COMPENSATED.
C       KFLAG: SAME AS KEIFLG IN PROCII.
C        LEVM: NUMBER OF LEVELS IN THE NETWORK (NOTE EX. VAR. ARE ALSO
C              ASSIGNED LEVELS JUST LIKE GATES).
C      LGLIST: LGLIST(J) TELLS NO. OF GATES AND EX. VAR. IN LEVEL J OF
C              NETWORK.
C         LIP: NUMBER OF PREDECESSORS FOR THE GATE UNDER CONSIDERATION.
C      LIPRED: LIPRED(GI) TELLS NO. OF IMMEDIATE PREDECESSORS OF GATE GI.
C       LISTC: ORDERED LIST OF CONNECTIBLE INPUTS TO GCO.  ORDERED BY
C              DECREASING NO. OF 0'S IN GCO COVERED.
C       LISTL: ORDERED LIST OF GATES AND EX. VAR. WHICH ORIGINALLY FED
C              GCO AND WHICH HAVE NOT YET BEEN DISCONNECTED.  ORDERED BY
C              DECREASING NO. OF ESSENTIAL 1'S.
C      LISUCC: LISUCC(GI) TELLS NO. OF IMMEDIATE SUCCESSORS OF GATE (OR
C              EX. VAR.) GI.
C       LMTS2: UPPER LIMIT OF THE NUMBER OF ELEMENTS IN SET S2.
C      LPOTAB: FOR GATE GI, LPOTAB(GI) POINTS TO LAST ROW OF POTAB
C              CONCERNING GI.
C           M: NUMBER OF NETWORK OUTPUT GATES.
C           N: NUMBER OF EXTERNAL VARIABLES (OR INPUT FNC.) AVAILABLE.
C      NEPMAX: FOR ERROR COMPENSATION PROGRAMS.  IF MORE THAN NEPMAX
C              ERROR POSITIONS OCCUR WHEN A PARTICULAR GATE IS REMOVED,
C              PROGRAM SKIPS ATTEMPT TO COMPENSATE FOR THAT GATE'S
C              REMOVAL.  VALUE CAN BE SPECIFIED BY USER, OTHERWISE EQUAL
C              TO ONE HALF OF N2 BY DEFAULT.
C          NM: SUM OF N PLUS M
C         NM1: SUM OF NM PLUS 1.
C         NN2: PRODUCT OF N AND N2.
C         NOS: NUMBER OF ELEMENTS IN SET S.
C        NOS1: NUMBER OF ELEMENTS IN SET S1.
C      NOS1SV: NUMBER OF ELEMENTS IN SET S1 BEFORE ENTERING SUBROUTINE
C              RPLCF.
C        NOS2: NUMBER OF ELEMENTS IN SET S2.
C        NOT1: NUMBER OF ELEMENTS IN SET T1.
C      NOT1SV: NUMBER OF ELEMENTS IN SET T1 BEFORE ENTERING SUBROUTINE
C              RPLCF.
C         NO0: NUMBER OF ACTIVE ELEMENTS IN ARRAY IDX0.
C        NO0E: NUMBER OF ACTIVE ELEMENTS IN ARRAY IDX0E.
C         NO1: NUMBER OF ACTIVE ELEMENTS IN ARRAY IDX1.
C        NO1E: NUMBER OF ACTIVE ELEMENTS IN ARRAY IDX1E.
C          NR: SUM OF N PLUS R.
C        NRN2: PRODUCT OF NR AND N2.
C       NRPLC: NRPLC(I) STORES THE NUMBER OF ELEMENTS IN RPLC(I,*)
C                                                     FOR I=1,2.
C          N1: SUM OF N PLUS 1.
C          N2: NUMBER OF DIFFERENT INPUT COMBINATIONS TO BE CONSIDERED
```

```
C             (USUALLY 2 TO THE POWER N).
C     ORIGIN: ORIGIN(GI)=1 MEANS GI ORIGINALLY CONNECTED TO GCO.
C             ORIGIN(GI)=0 MEANS GI DID NOT FEED GCO ORIGINALLY.
C         P$: P$(1,-) CONSECUTIVELY LISTS OUTPUTS OF EVERY EX. VAR. AND
C             EVERY GATE (FOR EVERY INPUT COMBINATION): P$(1,1),...,
C             P$(1,N2) FOR FIRST EX VAR; P$(1,N2+1),...,P$(1,2*N2) FOR
C             SECOND EX VAR; ... ; P$(1,N*N2+1),..., P$(1,N*N2+N2) FOR
C             FIRST GATE; ETC.  P$(2,-) IS USED AS WORK SPACE FOR
C             CALCULATIONS ASSOCIATED WITH P$(1,-).
C        PCO: FOR ERROR COMPENSATION PROCEDURES.  PCO IS THE GATE
C             REMOVED FROM ORIGINAL NETWORK TO OBTAIN CURRENT ALTERED
C             NETWORK.
C     POINTA: NOT USED.
C     POINTC: POINTS TO LAST ELEMENT IN LISTC.
C     POINTL: POINTS TO LAST ELEMENT IN LISTL.
C     POINTR: POINTS TO LAST ELEMENT IN RNEC1 (IN SUBROUTINE SUBSTI).
C      POTAB: POSSIBLE OUTPUT TABLE.  HOLDS INFORMATION ABOUT ALL
C             COMBINATIONS OF CONNECTIONS TO FORM NEW (AND HOPEFULLY
C             USEFUL) FUNCTIONS.
C     PPOTAB: FOR GATE GI, PPOTAB(GI) POINTS TO FIRST OF A SEQUENCE OF
C             ROWS OF POTAB CONCERNING GI.
C          R: NUMBER OF GATES IN THE NETWORK (EXCLUDES EX VAR, ALSO
C             NOTE SOME OF R GATES MAY BE ISOLATED).
C       RPLC: RPLC(1,*) STORES THE SELECTED GATE'S IP GATES WHICH HAVE
C                       ERROR-COORDINATES OF WEIGHT 2 OR ABOVE.
C             RPLC(2,*) STORES THE SELECTED GATE'S IP GATES WHICH HAVE
C                       AT LEAST ONE ERROR-COORDINATE OF WEIGHT 1.
C     RSCONN: LIST OF CONNECTIONS ADDED TO A NETWORK (IN CODED FORM).
C     RTCONN: LIST OF CONNECTIONS REMOVED FROM A NETWORK (CODED FORM).
C          S: NO. OF CONNECTIONS ADDED TO A NETWORK.  POINTS TO LAST
C             ENTRY IN RSCONN.
C       SETS: SET S CONSISTING OF INPUTS OF THE GATE UNDER CONSIDERATION
C             WHICH ARE TO BE REPLACED IF POSSIBLE.
C      SETS1: SET S1 CONSISTING OF ELEMENTS OF SET S WHICH CAN BE
C             REPLACED BY ELEMENTS IN SET S2.
C      SETS2: SET S2 CONSISTING OF FUNCTIONS WHICH ARE CANDIDATES FOR
C             REPLACING ELEMENTS IN SET S.
C      SETT1: SET T1 CONSISTING OF ESSENTIAL ONES COVERED BY ELEMENTS IN
C                                                                SET S1.
C        STS: STARTING ELEMENT OF SET S.
C     SUC$MX: SUC$MX(GI,GJ)>0 MEANS GATE GJ IS A SUCCESSOR OF GATE GI.
C             SUC$MX(GI,GJ)=0 IF NOT.
C       SUMP: SUM OF ALL ACTIVE INPUTS OF THE GATE UNDER CONSIDERATION.
C      SUMS2: SUM OF ALL ACTIVE ELEMENTS OF SET S2.
C          T: NUMBER OF CONNECTIONS REMOVED FROM A NETWORK.  POINTS TO
C             LAST ENTRY IN RTCONN.
C       TIME: USED TO STORE AMOUNT OF ELAPSED COMPUTATION TIME.
C      UNAME: MNEMONIC NAMES FOR EXTERNAL VARIABLES AND GATES.
C     VF$UB1: POINTS TO LAST ELEMENT IN VF$1.
C       VF$1: SIMILAR TO F$1, EXCEPT THIS LISTS JUST COMPONENT POSITIONS
C             (OF 0'S IN CSPF VECTOR OF GCO) COVERED ONLY BY REMAINING
C             ORIGINALLY CONNECTED INPUTS TO GCO.
C
C
C
      COMMON NEPMAX
      COMMON    N            , M             , A             , B
     1    ,     R            , N2            , N1            , NR
     2    ,     NM           , KFLAG         , JFLAG         , COST
     3    ,     LEVM         , NRN2          , NM1           , NN2
      COMMON    ISUCC(40,40) , LISUCC(40)    , IPRED(40,40)  , LIPRED(40)
```

```
     1      ,     INC$MX(40,40), SUC$MX(40,40), P$(2,1280)    , UNAME(40)    G3 01590
     2      ,     GLEVEL(40)     , LGLIST(40)     , HLIST(40,40)  , TIME         G3 01600
      COMMON      T              , RTCONN(100)    , S             , RSCONN(100)  G3 01610
      COMMON      IFLAG          ,POINTA          ,ESS1S(40)      ,F$1(32)       G3 01620
     1      ,F$UB1          ,INPTCV(32)      ,LISTC(40)      ,POINTC        G3 01630
     2      ,LISTL(40)      ,POINTL          ,ORIGIN(40)     ,IPATH(40)     G3 01640
     3      ,POINTR         ,VF$1(32)        ,VF$UB1         ,GSMALL(40,32)G3 01650
      COMMON      POTAB(200,42),PPOTAB(40)      ,LPOTAB(40)     ,NRPLC(2)      G3 01660
     1      ,RPLC(2,40)     ,IDX0(32)        ,IDX0E(32)      ,IDX1(32)      G3 01670
     2      ,IDX1E(32)      ,SUMP(32)        ,SETT1(32)      ,NOT1          G3 01680
     3      ,SETS1(40)      ,NOS1            ,SETS(40)       ,NOS           G3 01690
     4      ,STS            ,SUMS2(32)       ,SETS2(200)     ,NOS2          G3 01700
     5      ,LIP            ,NO0E            ,KEYA           ,KEYB          G3 01710
     6      ,NO0            ,NO1             ,NO1E           ,$GT           G3 01720
     7      ,$LTH           ,$PW             ,$NOE           ,GI            G3 01730
      COMMON                 NOT1SV         ,NOS1SV         ,LMTS2          G3 01740
      DIMENSION CNTLIS(144),UGATE(40),UHEAD(20)                               G3 01750
      DATA KOUNT5 /0/, UBLANK/'    '/                                         G3 01760
  990 READ(5,1000,END=500) UHEAD, N, M, R, A, B, UC, NEPMAX                   G3 01770
C     NEPMAX IS THE MAXIMUM ALLOWABLE NUMBER OF ERROR POSITIONS               G3 01780
 1000 FORMAT(20A4/5I4,A4,I4)                                                  G3 01790
      KEYXC=0                                                                 G3 01800
      IF(UC.NE.UBLANK) KEYXC=1                                                G3 01810
      CALL PAGE                                                               G3 01820
      CALL LINE(10)                                                           G3 01830
      KOUNT5=KOUNT5+1                                                         G3 01840
      PRINT 2, KOUNT5                                                         G3 01850
    2 FORMAT(20X,'*** OPTIMAL NOR NETWORK ***',50X,'PROBLEM NO.= ',I4 ) G3 01860
      CALL LINE(4)                                                            G3 01870
      PRINT 1005, UHEAD                                                       G3 01880
 1005 FORMAT(25X,20A4)                                                        G3 01890
      CALL LINE(4)                                                            G3 01900
      PRINT 10, N,M,A,B                                                       G3 01910
   10 FORMAT(30X,'NUMBER OF VARIABLES =',I4 //                                G3 01920
     1        30X,'NUMBER OF FUNCTIONS =',I4 //                               G3 01930
     2        30X,'COST COEFFICIENT A   =',I4//                               G3 01940
     3        47X,                  'B   =',I4)                               G3 01950
      CALL LINE(1)                                                            G3 01960
      IF(KEYXC.NE.0) GO TO 25                                                 G3 01970
      PRINT 21                                                                G3 01980
   21 FORMAT(1H0,29X,'--- UNCOMPLEMENTED VARIABLES  X ---')                   G3 01990
      GO TO 30                                                                G3 02000
   25 CONTINUE                                                                G3 02010
      PRINT 28                                                                G3 02020
   28 FORMAT(1H0,29X,'--- BOTH COMPLEMENTED AND UNCOMPLEMENTED VARIABLESG3 02030
     1 X, Y ---')                                                             G3 02040
   30 CONTINUE                                                                G3 02050
      CALL LINE(5)                                                            G3 02060
C***** SET UP EXTERNAL VARIABLES *****                                        G3 02070
      N2=2**N                                                                 G3 02080
      IF(NEPMAX.EQ.0)NEPMAX = N2/2                                            G3 02090
      H=N*N2                                                                  G3 02100
      J=N2                                                                    G3 02110
      L= 1                                                                    G3 02120
      I=0                                                                     G3 02130
      DO 1011 II=1,N                                                          G3 02140
       J=J/2                                                                  G3 02150
       L=L*2                                                                  G3 02160
       SN= 1                                                                  G3 02170
       DO 1010 LL=1,L                                                         G3 02180
        SN=-SN                                                                G3 02190
```

```
      V=(1+SN)/2
      DO 1009 JJ=1,J
       I=I+1
       P$(1,I)=V
     IF(KEYXC.NE.0)P$(1,I+H)=1-V
009    CONTINUE
010   CONTINUE
01'  CONTINUE
     IF(KEYXC.NE.0) N=N+N
     N1=N+1
     NM=N+M
     NM1=NM+1
     NN2=N*N2+1
     NR=N+R
     NRN2=NR*N2
     CALL OUTPUT(INC$MX,KEYXC)
**** READ IN NETWORK INFORMATION AND SET UP INC$MX *****
     READ 1001,    CNTLIS
001  FORMAT(16I5)
     DO 1115 GI=1,NR
     DO 1115 GJ=1,NR
115  INC$MX(GI,GJ)=0
     DO 1120 I=1,144
      ITEM=CNTLIS(I)
     IF(ITEM.EQ.0) GO TO 1119
      GI=ITEM/100
      GJ=ITEM-100*GI
      INC$MX(GI,GJ)=1
      GO TO 1120
119  COST=A*R+B*(I-1)
      GO TO 1130
120  CONTINUE
130  CONTINUE
     CALL SUBNET
     CALL PVALUE
     CALL LINE(4)
     PRINT 1140, COST
140  FORMAT(20X,' ORIGINAL NETWORK     COST=', I5)
     CALL LINE(4)
     CALL TRUTH(P$,1)
     CALL LINE(4)
     CALL CKT(INC$MX,GLEVEL)

**** ENTRY REDUNDANCY CHECK *****
     S = 0
     T = 0
     CALL UNNECE
     GATES = M
     C = 0
     DO 4 GI =  1,NR
     C = C + LISUCC(GI)
     IF(GI.LE.NM)GOTO4
     IF(LISUCC(GI).GT.0)GATES=GATES+1
   4 CONTINUE
     OLDCST = A*GATES + B*(C)
     T=0
     S=0
     INITIALIZE TIMER TO 10 MINUTES
     CALL STIMEZ(60000)
     TIME = KTIMEZ(0)
***    PROCEDURE  GTMERG
```

```
      CALL GTMERG
C     CALL FOR ELAPSED TIME
      TIME = KTIMEZ(0) - TIME
      CALL LINE(4)
      PRINT 3915
 3916 FORMAT(20X,'TIME ELAPSED =',I8,'  CENTISECONDS')
 3915 FORMAT(20X,'NETWORK DERIVED BY GTMERG')
      PRINT 3916,TIME
      CALL LINE(4)
      CALL TRUTH(P$,1)
      CALL LINE(4)
      CALL CKT(INC$MX,GLEVEL)
      GATES = M
      C = 0
      DO 36 GI = 1,NR
      C = C + LISUCC(GI)
      IF(GI.LE.NM) GO TO 36
      IF(LISUCC(GI).GT.0) GATES = GATES + 1
   36 CONTINUE
      NEWCST = A*GATES + B*C
      IF(NEWCST.LT.OLDCST)GO TO 37
      PRINT 105
  105 FORMAT(1H ,10X,'NO REDUNDANCY FOUND.')
      GO TO 990
   37 CALL LINE(3)
      PRINT 320,NEWCST
  320 FORMAT(9X,'* A NETWORK DERIVED BY GTMERG'/9X,' COST=',I5,'.')
      GO TO 990
  500 STOP
      END


      SUBROUTINE MINI2(IMPROV)
C     EDITION AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
C     THE NAME ATTEMPTS TO INDICATE THAT THIS SUBROUTINE IS A MINIATURE
C     VERSION OF PROCEDURE II (PROCII) - ACTUALLY, THIS ROUTINE ONLY
C     REMOVES CONNECTIONS, NONE ARE ADDED
      IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)
C
C     DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.
C
      COMMON NEPMAX
      COMMON    N            , M              , A              , B
     1     ,    R            , N2             , N1             , NR
     2     ,    NM           , KFLAG          , JFLAG          , COST
     3     ,    LEVM         , NRN2           , NM1            , NN2
      COMMON    ISUCC(40,40) , LISUCC(40)     , IPRED(40,40)   , LIPRED(40)
     1     ,    INC$MX(40,40), SUC$MX(40,40), P$(2,1280)       , UNAME(40)
     2     ,    GLEVEL(40)   , LGLIST(40)     , HLIST(40,40)   , TIME
      COMMON    T            , RTCONN(100)    , S              , RSCONN(100)
      COMMON    IFLAG        ,POINTA          ,ESS1S(40)       ,F$1(32)
     1      ,F$UB1           ,INPTCV(32)      ,LISTC(40)       ,POINTC
     2      ,LISTL(40)       ,POINTL          ,ORIGIN(40)      ,IPATH(40)
     3      ,POINTR          ,VF$1(32)        ,VF$UB1          ,GSMALL(40,32)
      COMMON    POTAB(200,42),PPOTAB(40)      ,LPOTAB(40)      ,NRPLC(2)
     1      ,RPLC(2,40)      ,IDX0(32)        ,IDX0E(32)       ,IDX1(32)
     2      ,IDX1E(32)       ,SUMP(32)        ,SETT1(32)       ,NOT1
     3      ,SETS1(40)       ,NOS1            ,SETS(40)        ,NOS
     4      ,STS             ,SUMS2(32)       ,SETS2(200)      ,NOS2
     5      ,LIP             ,NOOE            ,KEYA            ,KEYB
     6      ,NOO             ,NO1             ,NO1E            ,$GT
```

```
 7     ,$LTH                ,$PW             ,$NOE          ,GI
  COMMON                  NOT1SV           ,NOS1SV        ,LMTS2
  DIMENSION T1PRED(40),T2PRED(40),GORDER(40),F$0(32),MARKED(40)
  DIMENSION USED(40),TORDER(40)
  IMPROV = 0
  T = 0
  ORDER GATES IN GORDER
  EFLAG = 0
  GO TO 63
  THIS ENTRY POINT FOR CALCULATION OF GORDER ONLY
  ENTRY FORMGO
  EFLAG = 1
63 CONTINUE
  COUNT = 0
  DO 1 I=1,LEVM
  NMINLV = LGLIST(I)
  IF(NMINLV.EQ.0)GOTO1
  DO 2 J=1,NMINLV
  COUNT = COUNT + 1
  GORDER(COUNT) = HLIST(J,I)
 2 CONTINUE
 1 CONTINUE
  IF(EFLAG.EQ.1)RETURN
  CALCULATE NUMBER OF OUTPUTS OF EACH GATE
  (THE ARRAY 'USED' IS USED HERE JUST TEMPORARILY)
  DO 51 I=N1,NR
  TCOUNT = 0
  DO 52 J=1,NR
  IF(INC$MX(I,J).EQ.1)TCOUNT = TCOUNT + 1
52 CONTINUE
  TCOUNT NOW CONTAINS  THE NUMBER OF OUTPUTS OF GATE I
  USED(I) = TCOUNT
51 CONTINUE
  MOST = 0
  DO 53 I =N1,NR
  IF(USED(I).GT.MOST)MOST = USED(I)
53 CONTINUE
  DO 56 I= 1,N
56 TORDER(I) = I
  TPOINT = N1
  MOST = MOST + 1
50 MOST = MOST - 1
  IF(MOST.LT.0)GO TO 54
  DO 55 I=1,NR
  II = GORDER(I)
  IF(II.LE.N)GO TO 55
  IF(USED(II).NE.MOST)GO TO 55
  TORDER(TPOINT) = II
  TPOINT = TPOINT + 1
55 CONTINUE
  GO TO 50
54 CONTINUE
  INITIALIZE GSMALL
  DO 4 I=N1,NM
  X = (I-1)*N2
  DO 4 J=1,N2
  Y = P$(1,X+J)
  IF(Y.EQ.0)GSMALL(I,J) = -100
  IF(Y.EQ.1)GSMALL(I,J) = 1
  IF(Y.EQ.-1)GSMALL(I,J)=0
 4 CONTINUE
```

```
      EFLAG = 0
      GO TO 57
      ENTRY INITGS
      EFLAG = 1
   57 DO 3 I=1,NR
      USED(I) = 0
      IF(I.LT.N1)GO TO 58
      IF(I.GT.NM) GO TO 58
      GO TO 3
   58 DO 59 J = 1,N2
   59 GSMALL(I,J)= 0
    3 CONTINUE
      DO 62 I = N1,NM
      USED(I) = 1
   62 CONTINUE
C     INITIALIZATION
      DO 34 I=1,NP
      GATE = GORDER(I)
      IF(GATE.LT.N1)GO TO 34
      XX= LIPRED(GATE)
      IF(XX.EQ.0)GOTO34
      F$UB1 = 0
      F$UBO = 0
      DO 35 J=1,N2
      COMPNT = GSMALL(GATE,J)
      IF(COMPNT.EQ.0)GO TO 35
      IF(COMPNT.LT.0)GO TO 36
      IF(COMPNT.GE.1000) GO TO 35
      F$UBO = F$UBO + 1
      F$O(F$UBO) = J
      GO TO 35
   36 IF(COMPNT.LE.-1000) GO TO 35
      F$UB1 = F$UB1 + 1
      F$1(F$UB1) = J
   35 CONTINUE
      IF(F$UB1.EQ.0)GO TO 34
      DO 38 K=1,XX
      FEEDGT = IPRED(K,GATE)
      X = (FEEDGT-1)*N2
      DO 39 L=1,F$UB1
      Y = F$1(L)
      IF(P$(1,X+Y).LE.0)GO TO 39
      IF(GSMALL(FEEDGT,Y).GT.1000)GOTO39
      IF(GSMALL(GATE,Y).EQ.-200)GOTO39
      IF(GSMALL(GATE,Y).EQ.-100)GO TO 40
      GSMALL(GATE,Y) = -200
      GO TO 39
   40 GSMALL(GATE,Y) = -FEEDGT
   39 CONTINUE
   38 CONTINUE
      DO 60 K=1,XX
   60 MARKED(IPRED(K,GATE)) = 0
      DO 41 K=1,F$UB1
      X = GSMALL(GATE,F$1(K))
      IF(X.EQ.-100)GO TO 41
      IF(X.EQ.-200)GOTO41
      X = -X
      GSMALL(+X,F$1(K))=1
      USED(X) = 1
      IF(MARKED(X).EQ.1)GOTO41
      MARKED(X) = 1
```

```
   DO 42 L=1,F$UBO
   Y = GSMALL(X,F$O(L))
   IF(Y.GT.1000.OR.Y.LT.-1000)GO TO 42
   GSMALL(+X,F$O(L))=-100
42 CONTINUE
41 CONTINUE
34 CONTINUE
   IF(EFLAG.EQ.1)RETURN
   INITIALIZE COUNTER TO LOOP ONCE FOR EACH GATE
   GCOUNT = 0
   INCREMENT GCOUNT
 5 GCOUNT = GCOUNT + 1
   ARE ALL GATES EXHAUSTED?
   IF(GCOUNT.LE.NR)GO TO 6
   IF(T.GT.0) IMPROV = 1
   IF(IMPROV.EQ.0)RETURN
   IF HERE, NETWORK WAS ALTERED, SO UPDATE ARRAYS
   CALL SUBNET
   CALL PVALUE
   RETURN
 6 GCO = GORDER(GCOUNT)
   IS GCO AN ISOLATED GATE OR EXTERNAL VARIABLE?
   IF(GCO.LE.N)GOTO5
   DO 8 I=1,N2
   IF(GSMALL(GCO,I).GE.1)GOTO7
 8 CONTINUE
   IF HERE, GATE IS ISOLATED - REMOVE INPUTS
   X = LIPRED(GCO)
   IF(X.EQ.0)GOTO5
   DO 9 I=1,X
   Y = IPRED(I,GCO)
   INC$MX(Y,GCO) = 0
   RECORD THE DISCONNECTION
   T = T + 1
 9 CONTINUE
   GOTO 5
   REMOVE UNNECESSARY CONNECTIONS TO GCO IN THE NEXT FEW SECTIONS

   CALCULATE F(GCO)
 7 F$UB1 = 0
   DO 10 I=1,N2
   IF(GSMALL(GCO,I).GE.0)GOTO10
   F$UB1 = F$UB1 + 1
   F$1(F$UB1) = I
10 CONTINUE
   DO 11 I=1,F$UB1
11 INPTCV(F$1(I)) = 0
   X = LIPRED(GCO)
   DO 222I=1,X
   ESSIS(IPRED(I,GCO)) = 0
222 CONTINUE
   T1SUB = 0
   T2SUB = 0
   DO 48 I = 1,NR
   IF(INC$MX(I,GCO).EQ.0)GOTO48
   T1SUB = T1SUB + 1
   T1PRED(T1SUB) = I
48 CONTINUE
17 DO 18 I=1,X
   Y = (T1PRED(I)-1)*N2
   DO 19 J=1,F$UB1
```

```
      Q = F$1(J)
      IF(P$(1,Y+Q).NE.1)GO TO 19
      IF(INPTCV(Q).LE.0) GO TO 20
      INPTCV(Q) = INPTCV(Q) + 1
      GO TO 19
   20 IF(INPTCV(Q).LT.0)GO TO 21
      INPTCV(Q) = -T1PRED(I)
      GO TO 19
   21 INPTCV(Q) = 2
   19 CONTINUE
   18 CONTINUE
C     MARK ESSENTIAL 1'S
      DO 22 I=1,F$UB1
      Q = INPTCV(F$1(I))
      IF(Q.GE.0)GO TO 22
      ESS1S(-Q) = ESS1S(-Q) + 1
   22 CONTINUE
   46 SELECT = 0
      BESTSL = 0
      DO 45 L=1,X
      Q = T1PRED(L)
      IF(INC$MX(Q,GCO).EQ.0)GOTO45
      IF(ESS1S(Q).GT.0)GOTO45
      IF(SELECT.EQ.0)SELECT = Q
      IF(USED(Q).EQ.1)GOTO45
      IF(BESTSL.NE.0)GOTO45
      BESTSL = Q
   45 CONTINUE
      IF(SELECT.EQ.0)GO TO 47
      Q = SELECT
      IF(BESTSL.NE.0)Q = BESTSL
C     IF HERE, GATE HAS NO ESSENTIAL 1'S  -  REMOVE IT
      INC$MX(Q,GCO) = 0
      T = T + 1
C     UPDATE ESS1S
      Y = (Q - 1)*N2
      DO 24 J=1,F$UB1
      V = F$1(J)
      IF(P$(1,Y+V).NE.1)GO TO 24
C     UPDATE INPTCV FOR COMPONENT V
      INPTCV(V) = INPTCV(V) - 1
      IF(INPTCV(V).GT.1)GO TO 24
C     CASE WHEN NEW ESSEN 1 CREATED
      DO 27 K = 1,X
      W = T1PRED(K)
      IF(INC$MX(W,GCO).EQ.0) GO TO 27
      Z = (W - 1) * N2
      IF(P$(1,Z+V).EQ.0)GO TO 27
      ESS1S(W) = ESS1S(W) + 1
C     IN THIS CASE, NO NEED TO UPDATE INPTCV FURTHER
      GSMALL(GCO,V) = -W
      GO TO 24
   27 CONTINUE
   24 CONTINUE
      GOTO46
   47 DO 49 I = 1,NR
      IF(INC$MX(I,GCO).EQ.0)GOTO49
      T2SUB = T2SUB + 1
      T2PRED(T2SUB) = I
   49 CONTINUE
C     NOW ALL CURRENT INPUTS HAVE ESSENTIAL 1'S
```

```
   INPUTS STILL CONNECTED TO GCO ARE LISTED IN T2PRED IN REVERSE
   ORDER

   UPDATE G(I)'S OF THOSE GATES STILL CONNECTED TO GATE GCO

   DO 29 II=1,F$UB1
   I = F$1(II)
   CHOICE = -GSMALL(GCO,I)
   IF(CHOICE.LT.100)GO TO 61
   CHOICE = 0
   DO 30 JJJ=1,NR
   JJ = TORDER(JJJ)
   IF(INC$MX(JJ,GCO).EQ.0)GO TO 30
   IF(P$(1,(JJ-1)*N2+I).NE.1)GO TO 30
   IF(JJ.LE.N)GO TO 29
   IF(CHOICE.EQ.0)CHOICE=JJ
   IF(GSMALL(JJ,I).GE.1)GOTO29
30 CONTINUE
61 GSMALL(CHOICE,I) = 1
   USED(CHOICE) = 1
29 CONTINUE
   DO 32 I=1,N2
   IF(GSMALL(GCO,I).LT.1)GO TO 32
   DO 33 J=1,T2SUB
   IF(GSMALL(T2PRED(J),I).EQ.0)GSMALL(T2PRED(J),I)=-100
33 CONTINUE
32 CONTINUE
   GOTO5
   END


   SUBROUTINE GTMERG
   IMPLICIT INTEGER*4(A-T,V-Z), REAL(U)
**** THIS SUBROUTINE TRIES TO MERGE TWO GATES, THE INTERSECTION OF
   WHOSE CSPF'S IS NOT EMPTY, TO ONE WHICH IS INSIDE THE INTERSECTION
   OF THEIR CSPF'S ******

   DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.

   COMMON NEPMAX
   COMMON     N             , M              , A              , B
  1    ,      R             , N2             , N1             , NR
  2    ,      NM            , KFLAG          , JFLAG          , COST
  3    ,      LEVM          , NRN2           , NM1            , NN2
   COMMON     ISUCC(40,40)  , LISUCC(40)     , IPRED(40,40)   , LIPRED(40)
  1    ,      INC$MX(40,40) , SUC$MX(40,40)  , P$(2,1280)     , UNAME(40)
  2    ,      GLEVEL(40)    , LGLIST(40)     , HLIST(40,40)   , TIME
   COMMON     T             , RTCONN(100)    , S              , RSCONN(100)
   COMMON     IFLAG         ,POINTA          ,ESS1S(40)       ,F$1(32)
  1     ,F$UB1              ,INPTCV(32)      ,LISTC(40)       ,POINTC
  2     ,LISTL(40)          ,POINTL          ,ORIGIN(40)      ,IPATH(40)
  3     ,POINTR             ,VF$1(32)        ,VF$UB1          ,GSMALL(40,32)
   COMMON     POTAB(200,42),PPOTAB(40)       ,LPOTAB(40)      ,NRPLC(2)
  1     ,RPLC(2,40)         ,IDX0(32)        ,IDX0E(32)       ,IDX1(32)
  2     ,IDX1E(32)          ,SUMP(32)        ,SETT1(32)       ,NOT1
  3     ,SETS1(40)          ,NOS1            ,SETS(40)        ,NOS
  4     ,STS                ,SUMS2(32)       ,SETS2(200)      ,NOS2
  5     ,LIP                ,NO0E            ,KEYA            ,KEYB
  6     ,NO0                ,NO1             ,NO1E            ,$GT
  7     ,$LTH               ,$PW             ,$NOE            ,GI$$$$
   COMMON                   NOT1SV          ,NOS1SV          ,LMTS2
```

```
      DIMENSION INDEXO(32),INDEX1(32),LICAND(40)
    1 CONTINUE
      CALL MINI2(IMPROV)
      NR1=NR-1
      DO 59 GI=NM1,NR1
       IF(GLEVEL(GI).EQ.1) GO TO 59
       BSGI=(GI-1)*N2
       GI1=GI+1
       DO 49 GJ=GI1,NR
        IF(GLEVEL(GJ).EQ.1 .OR. INC$MX(GI,GJ).GE.1
     1      .OR. INC$MX(GJ,GI).GE.1) GO TO 49
       KGIGJ=0
       KGJGI=0
       NOO=0
       NO1=0
        BSGJ=(GJ-1)*N2
        DO 19 TH=1,N2
         IF(GSMALL(GI,TH)) 5,10,15
C**** TH OF GI IS A'0' *****
    5    IF(GSMALL(GJ,TH)) 6,7,8
C**** TH OF GI AND GJ ARE BOTH '0' *****
    6    NOO=NOO+1
         INDEXO(NOO)=TH
         GO TO 19
C**** TH OF GI IS A '0' AND TH OF GJ IS A '*' *****
    7    IF(P$(1,BSGJ+TH).EQ.0) GO TO 6
C***** GJ CAN NOT SUBSTITUTE FOR GI *****
         KGJGI=1
         GO TO 6
C***** GI, GJ CAN NOT BE MERGED TO ONE GATE *****
    8    GO TO 49
C
C***** GSMALL(GI,TH) IS A DON'T CARE *****
C
   10    IF(GSMALL(GJ,TH)) 11, 12, 13
C***** TH OF GJ IS A '0' *****
   11    IF(P$(1,BSGI+TH).EQ.0) GO TO 6
         KGIGJ=1
         GO TO 6
C***** BOTH GI AND GJ ARE DON'T CARES *****
   12    GO TO 19
C***** CORRESPONDING COMPONENT OF GJ IS '1' *****
   13    IF(P$(1,BSGI+TH).EQ.1) GO TO 18
         KGIGJ=1
         GO TO 18
C*****  GSMALL(GI,TH) IS A '1' *****
   15    IF(GSMALL(GJ,TH)) 16, 17, 18
C***** GI AND GJ CAN NOT BE MERGED TO ONE GATE *****
   16   GO TO 49
C***** GSMALL(GJ,TH) IS A DON'T CARE *****
   17    IF(P$(1,BSGJ+TH).EQ.1) GO TO 18
C***** GJ CAN NOT SUBSTITUTE FOR GI *****
         KGJGI=1
C***** BOTH GSMALL(GI,TH) AND GSMALL(GJ,TH) ARE '1' *****
   18    NO1=NO1+1
         INDEX1(NO1)=TH
   19   CONTINUE
C
C***** GI AND GJ MAY BE MERGED *****
        IF (KGIGJ.EQ.0.AND.SUC$MX(GJ,GI).LE.0) GO TO 65
        IF(KGJGI.EQ.0 .AND. SUC$MX(GI,GJ).LE.0) GO TO 60
```

```
C***** TRY TO BUILD UP A NEW GATE TO REPLACE GI AND GJ *****
        CANDDT=0
       DO 29 GK=1,NR
        IF(GLEVEL(GK).EQ.1.AND.GK.GT.NM) GO TO 29
        IF(SUC$MX(GI,GK).GT.0.OR.SUC$MX(GJ,GK).GT.0) GO TO 29
C***** CHECK '1' COORDINATES(CORRESPONDING INPUT COORDINATE SHOULD BE 0)
        BSGK=(GK-1)*N2
C*****  IF(NOO.EQ.0) GO TO (GI,GJ ARE REDUNDANT)
C*****  IF(NO1.EQ.0) GO TO (GI,GJ AND GATES FED BY GI,GJ ARE REDUNDANT)
        DO 23 NRUN=1,NO1
         IF(P$(1,BSGK+INDEX1(NRUN)).EQ.1) GO TO 29
   23   CONTINUE
C***** GK IS CONNECTIBLE TO THE NEW GATE *****
        CANDDT=CANDDT+1
        LICAND(CANDDT)=GK
   29   CONTINUE
       IF(CANDDT.EQ.0) GO TO 49
       DO 30 NRUN=1,NOO
   30  SUMP(NRUN)=0
       DO 35 CAND=1,CANDDT
        GK=LICAND(CAND)
        BSGK=(GK-1)*N2
        DO 33 NRUN=1,NOO
   33   SUMP(NRUN)=SUMP(NRUN)+P$(1,BSGK+INDEXO(NRUN))
   35   CONTINUE
C
C***** CHECK IF '0' COORDINATES ARE COVERED *****
C
       DO 36 NRUN=1,NOO
        IF(SUMP(NRUN).LE.0) GO TO 49
   36   CONTINUE
C***** A NEW GATE CAN BE CONSTRUCTED TO REPLACE GI AND GJ *****
C***** MODIFY GI TO BE THE NEW GATE *****
   70   LIP=LIPRED(GI)
       DO 80 LI=1,LIP
        T=T+1
     INC$MX(IPRED(LI,GI),GI)=0
   80   CONTINUE
       DO 81 CAND=1,CANDDT
        S=S+1
     INC$MX(LICAND(CAND),GI)=1
   81   CONTINUE
C***** CONNECT GI TO SUCCESSORS OF GJ *****
   82   LIS=LISUCC(GJ)
       DO 85 LI=1,LIS
        GK=ISUCC(LI,GJ)
        T=T+1
     INC$MX(GJ,GK)=0
        S=S+1
     INC$MX(GI,GK)=1
   85   CONTINUE
       CALL SUBNET
       CALL UNNECE
        CALL PVALUE
C***** 'CALL RDTCNT' IS NOT  NECESSARY SINCE 'MINI2' WILL DO IT *****
       GO TO 1
   60  GK=GJ
       GJ=GI
       GI=GK
       GO TO 82
   65 CONTINUE
```

```
      GO TO 82
   49 CONTINUE
   59 CONTINUE
        RETURN
      END


      SUBROUTINE SUBNET
      IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)
C
C     DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.
C
      COMMON NEPMAX
      COMMON   N              , M               , A              , B
     1     ,   R              , N2              , N1             , NR
     2     ,   NM             , KFLAG           , JFLAG          , COST
     3     ,   LEVM           , NRN2            , NM1            , NN2
      COMMON   ISUCC(40,40)   , LISUCC(40)      , IPRED(40,40)   , LIPRED(40)
     1     ,   INC$MX(40,40)  , SUC$MX(40,40)   , P$(2,1280)     , UNAME(40)
     2     ,   GLEVEL(40)     , LGLIST(40)      , HLIST(40,40)   , TIME
      COMMON   T              , RTCONN(100)     , S              , RSCONN(100)
      COMMON   IFLAG          ,POINTA           ,ESSIS(40)       ,F$1(32)
     1     ,F$UB1             ,INPTCV(32)       ,LISTC(40)       ,POINTC
     2     ,LISTL(40)         ,POINTL           ,ORIGIN(40)      ,IPATH(40)
     3     ,POINTR            ,VF$1(32)         ,VF$UB1          ,GSMALL(40,32)
      COMMON   POTAB(200,42)  ,PPOTAB(40)       ,LPOTAB(40)      ,NRPLC(2)
     1     ,RPLC(2,40)        ,IDX0(32)         ,IDX0E(32)       ,IDX1(32)
     2     ,IDX1E(32)         ,SUMP(32)         ,SETT1(32)       ,NOT1
     3     ,SETS1(40)         ,NOS1             ,SETS(40)        ,NOS
     4     ,STS               ,SUMS2(32)        ,SETS2(200)      ,NOS2
     5     ,LIP               ,NOOE             ,KEYA            ,KEYB
     6     ,NOO               ,NO1              ,NO1E            ,$GT
     7     ,$LTH              ,$PW              ,$NOE            ,G$$$$$
      COMMON                   NOT1SV           ,NOS1SV          ,LMTS2
      DIMENSION X(40),LX(40,2),OUTO(40)
C     ENTRY PRESUC
    1 CONTINUE
      DO 10 GI=1,NR
       LS=0
       LP=0
       DO 5 GJ=1,NR
        IF(INC$MX(GI,GJ).EQ.0) GO TO 3
         LS=LS+1
         ISUCC(LS,GI)=GJ
         GO TO 5
    3   IF(INC$MX(GJ,GI).EQ.0) GO TO 5
         LP=LP+1
         IPRED(LP,GI)=GJ
    5  CONTINUE
       LISUCC(GI)=LS
       LIPRED(GI)=LP
   10 CONTINUE
C
      ENTRY SUCCES
      DO 21 GI=1,NR
      DO 21 GJ=1,NR
       SUC$MX(GI,GJ)=0
   21 CONTINUE
      DO 30 GJ=N1,NR
      DO 22 GS=1,NR
        X(GS)=0
```

```
 22  CONTINUE
     X(GJ)=1
     LO=1
     LX(1,1)=GJ
     V=1
 23  CONTINUE
     V=1-V
     SWO=1+V
     SW1=2-V
     L1=0
     DO 28 LL=1,LO
      GM=LX(LL,SWO)
      LIP=LIPRED(GM)
      IF(LIP.EQ.0) GO TO 28
      DO 26 LP=1,LIP
       GP=IPRED(LP,GM)
       IF(X(GP).GT.0) GO TO 26
        SUC$MX(GP,GJ)=1
        L1=L1+1
        LX(L1,SW1)=GP
        X(GP)=1
 26    CONTINUE
 28  CONTINUE
     IF(L1.EQ.0) GO TO 30
     LO=L1
     GO TO 23
 30 CONTINUE
C
C
    ENTRY LEVEL
    DO 40 GJ=1,NR
     OUTO(GJ)=LISUCC(GJ)
     GLEVEL(GJ)=-1
 40  CONTINUE
    LEV=0
 45 LEV=LEV+1
    G=0
    DO 50 GJ=1,NR
     IF(OUTO(GJ).GT.0 .OR. GLEVEL(GJ).GT.0) GO TO 50
     G=G+1
     HLIST(G,LEV)=GJ
     GLEVEL(GJ)=LEV
 50 CONTINUE
    IF(G.EQ.0) RETURN
    LGLIST(LEV)=G
    DO 60 GG=1,G
     GJ=HLIST(GG,LEV)
     LIP=LIPRED(GJ)
     IF(LIP.EQ.0) GO TO 60
     DO 55 LP=1,LIP
      GP=IPRED(LP,GJ)
      OUTO(GP)=OUTO(GP)-1
 55  CONTINUE
 60 CONTINUE
    LEVM=LEV
    GO TO 45
C
C
C
    ENTRY PVALUE
    DO 100 L=NN2,NRN2
     P$(1,L)=1
```

```
  100 CONTINUE
C
      LEV=LEVM
  110 CONTINUE
      LO=LGLIST(LEV)
      DO 130 L=1,LO
       GI=HLIST(L,LEV)
       LIS=LISUCC(GI)
       BSGI=(GI-1)*N2
       LJTH=0
       DO 115 JTH=1,N2
        IF(P$(1,BSGI+JTH).EQ.0) GO TO 115
        LJTH=LJTH+1
        X(LJTH)=JTH
  115  CONTINUE
       IF(LJTH.EQ.0) GO TO 130
       DO 125 LS=1,LIS
        GS=ISUCC(LS,GI)
        BSGS=(GS-1)*N2
        DO 120 LJ=1,LJTH
         P$(1,X(LJ)+BSGS)=0
  120   CONTINUE
  125  CONTINUE
  130 CONTINUE
      LEV=LEV-1
      IF(LEV.GE.2) GO TO 110
      RETURN
C
C
C
      ENTRY RSTRCT(KEYRST)
      KEYRST=0
      IF(LEVM.GT.LMAX)GO TO 160
      DO 150 GI=N1,NR
       IF(LIPRED(GI).GT.FANIN)GO TO 160
       IF(LISUCC(GI).GT.FANOUT)GO TO 160
  150 CONTINUE
      RETURN
  160 KEYRST=1
      RETURN
      ENTRY UNNECE
C***** THIS ENTRY DISCONNECT ALL GATES FROM WHICH THERE IS NO PATH
C        TO OUTPUT GATES *****
      TS=T
      DO 209 GI=NM1,NR
       IF(GLEVEL(GI).EQ.1) GO TO 207
       DO 205 GJ=N1,NM
        IF(SUC$MX(GI,GJ).GT.0) GO TO 209
  205  CONTINUE
C***** GI IS REDUNDANT *****
  207 CONTINUE
       LIP=LIPRED(GI)
       IF(LIP.EQ.0) GO TO 206
       DO 203 LI=1,LIP
        GK=IPRED(LI,GI)
      IF(INC$MX(GK,GI).LE.0) GO TO 203
        T=T+1
        RTCONN(T)=100*GK+GI
        INC$MX(GK,GI)=0
  203  CONTINUE
  206  LIS=LISUCC(GI)
```

```
      IF(LIS.EQ.0) GO TO 209
      DO 204 LI=1,LIS
       GK=ISUCC(LI,GI)
      IF(INC$MX(GI,GK).LE.0) GO TO 204
       T=T+1
       RTCONN(T)=100*GI+GK
       INC$MX(GI,GK)=0
  204  CONTINUE
  209 CONTINUE
      IF(T.GT.TS) GO TO 1
      RETURN
      END


      SUBROUTINE OUTPUT(MATRIX,ARRAY)
      IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)
C
C     DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.
C
      COMMON NEPMAX
      COMMON    N              , M              , A              , B
     1      ,   R              , N2             , N1             , NR
     2      ,   NM             , KFLAG          , JFLAG          , COST
     3      ,   LEVM           , NRN2           , NM1            , NN2
      COMMON    ISUCC(40,40)   , LISUCC(40)     , IPRED(40,40)   , LIPRED(40)
     1      ,   INC$MX(40,40)  , SUC$MX(40,40)  , P$(2,1280)     , UNAME(40)
     2      ,   GLEVEL(40)     , LGLIST(40)     , HLIST(40,40)   , TIME
      COMMON    T              , RTCONN(100)    , S              , RSCONN(100)
      COMMON    IFLAG          ,POINTA          ,ESS1S(40)       ,F$1(32)
     1      ,F$UB1             ,INPTCV(32)      ,LISTC(40)       ,POINTC
     2      ,LISTL(40)         ,POINTL          ,ORIGIN(40)      ,IPATH(40)
     3      ,POINTR            ,VF$1(32)        ,VF$UB1          ,GSMALL(40,32)
      COMMON    POTAB(200,42)  ,PPOTAB(40)      ,LPOTAB(40)      ,NRPLC(2)
     1      ,RPLC(2,40)        ,IDX0(32)        ,IDX0E(32)       ,IDX1(32)
     2      ,IDX1E(32)         ,SUMP(32)        ,SETT1(32)       ,NOT1
     3      ,SETS1(40)         ,NOS1            ,SETS(40)        ,NOS
     4      ,STS               ,SUMS2(32)       ,SETS2(200)      ,NOS2
     5      ,LIP               ,NOOE            ,KEYA            ,KEYB
     6      ,NO0               ,NO1             ,NO1E            ,$GT
     7      ,$LTH              ,$PW             ,$NOE            ,GI
      COMMON                    NOT1SV          ,NOS1SV          ,LMTS2
      DIMENSION UX(5), UY(5), UG(40), UF(40), ARRAY(40), ARRAY2(2,1280)
      DIMENSION MATRIX(40,40)
      DATA UX /' X1',' X2',' X3',' X4',' X5'/
      DATA UY/' Y1',' Y2',' Y3',' Y4',' Y5'/
      DATA UF /'  1','  2','  3','  4','  5','  6','  7','  8'
     1        , '  9',' 10',' 11',' 12',' 13',' 14',' 15',' 16'
     2        , ' 17',' 18',' 19',' 20',' 21',' 22',' 23',' 24'
     3        , ' 25',' 26',' 27',' 28',' 29',' 30',' 31',' 32'
     4        , ' 33',' 34',' 35',' 36',' 37',' 38',' 39',' 40'/
      DATA GMAX/40/

      KEYXC=ARRAY(1)
      IF(KEYXC.NE.0) GO TO 50
      DO 1 GI=1,N
       UNAME(GI)=UX(GI)
    1 CONTINUE
      GO TO 100
   50 CONTINUE
      L=N/2
      DO 4 GI=1,L
```

```
       UNAME(GI)=UX(GI)
      UNAME(GI+L)=UY(GI)
    4 CONTINUE
  100 CONTINUE
      DO 2 GI=N1,GMAX
       UNAME(GI)=UF(GI-N)
    2 CONTINUE
      RETURN
C
      ENTRY LINE(L)
      DO 6 LL=1,L
       PRINT 5
    5 FORMAT(1H )
    6 CONTINUE
      RETURN
C
      ENTRY PAGE
      PRINT 7
    7 FORMAT(1H1)
      RETURN
C
      ENTRY CKT(MATRIX,ARRAY)
      PRINT 10
   10 FORMAT(1H ,8X,'GATE .. LEVEL',6X, 'FED BY'/)
      DO 20 GJ=N1,NR
       G=0
       DO 15 GI=1,NR
        IF(MATRIX(GI,GJ).EQ.0) GO TO 15
        G=G+1
        UG(G)=UNAME(GI)
   15  CONTINUE
       IF(G.EQ.0) GO TO 18
       PRINT 17, UNAME(GJ),ARRAY(GJ),(UG(GG),GG=1,G)
   17  FORMAT(1H0, 9X,A3,5X,'/',I2,'/',5X,35(   A3))
       GO TO 20
   18  PRINT 19, UNAME(GJ),ARRAY(GJ)
   19  FORMAT(1H0, 9X,A3,5X,'/',I2,'/')
   20 CONTINUE
      RETURN
C
      ENTRY TRUTH(ARRAY2,J)
      IF(J.EQ.2) GO TO 36
      PRINT 35
   35 FORMAT(11X,   'TRUTH TABLE'/)
      GO TO 38
   36 PRINT 37
   37 FORMAT(11X,  'REQUIREMENT TABLE')
   38 CONTINUE
      DO 40 GI=1,NR
       ILO=(GI-1)*N2+1
       IHI=ILO+N2-1
       PRINT 41, UNAME(GI), (ARRAY2(J,I),I=ILO,IHI)
   40 CONTINUE
   41 FORMAT(1H0, 9X,A3,' = ', 32(I1,1X))
      RETURN
      END
```

```
C
C
C*****************************************************************************
C                                                                            *
C                                                                            *
C        PPPP    RRRR     OOO     GGG    RRRR      A      M     M            *
C        P   P   R   R   O   O   G   G   R   R    A A     MM   MM            *
C        P   P   R   R   O   O   G       R   R   A   A    M M M M            *
C        PPPP    RRRR    O   O   G  GG   RRRR    AAAAA    M  M  M            *
C        P       R  R    O   O   G   G   R  R    A   A    M     M            *
C        P       R   R    OOO     GGG    R   R   A   A    M     M            *
C                                                                            *
C                                                                            *
C                                                                            *
C N   N  EEEEE   TTTTT   TTTTT   RRRR      A              GGG       4        *
C NN  N  E         T       T     R   R    A A            G   G     44        *
C N N N  E         T       T     R   R   A   A           G        4 4        *
C N  NN  EEE       T       T     RRRR    AAAAA   XXXXX   G  GG   4  4        *
C N   N  E         T       T     R  R    A   A           G   G   44444       *
C N   N  EEEEE     T       T     R   R   A   A            GGG       4        *
C                                                                            *
C                                                                            *
C*****************************************************************************
C
C
      IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)
C     EDITION BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
C     NOTE: ALL COMMON VARIBLES MIGHT NOT BE USED IN THIS PROGRAM.
C
C     COMMON VARIABLES:
C         $GT: POINTS TO A 'COLUMN' OF POTAB. FOR EACH 'ROW' THE ENTRY
C              IN THIS COL. TELLS GATE WHERE FN. IS REALIZED.
C        $LTH: POINTS TO A 'COLUMN' OF POTAB.  FOR EACH 'ROW' THE ENTRY
C              IN THIS COL. TELLS HOW MANY CONNECTIONS MUST BE ADDED.
C        $NOE: POINTS TO A 'COLUMN' OF POTAB.  FOR EACH 'ROW' THE ENTRY
C              IN THIS COL. TELLS THE NUMBER OF 1-ERRORS CREATED IF THIS
C              ROW IS USED.
C         $PW: POINTS TO A 'COLUMN' OF POTAB.  FOR EACH 'ROW' THE ENTRY
C              IN THIS COLUMN TELLS THE PREFERENCE WEIGHT.
C           A: WEIGHT FOR NO. OF GATES IN COMPUTING COST FUNCTION.
C           B: WEIGHT FOR NO. OF CONNECTIONS IN COMPUTING COST FUNCTION.
C        COST: COST OF NETWORK - A MEASURE OF NETWORK SIZE.
C      ESS1S: RECORDS NO. OF ESSENTIAL 1'S IN EVERY INPUT TO CURRENT GCO
C              (POSITIONS IN ESS1S CORRES. TO GATES NOT FEEDING GCO ARE
C              IGNORED).
C       F$UB1: POINTS TO LAST ELEMENT IN F$1.
C         F$1: LISTS (CONSECUTIVELY) POSITIONS OF DESIRABLE 1'S (FOR
C              COVERING) IN A CONNECTIBLE FUNCTION.
C          GI: LABEL OF A PARTICULAR GATE.
C      GLEVEL: GLEVEL(GI) TELLS WHICH LEVEL OF THE NETWORK GI IS IN.
C      GSMALL: STORES INTERMEDIATE AND FINAL CALCULATED CSPF'S.
C       HLIST: HLIST(I,J) GIVES NAME OF I-TH GATE (OR EX. VAR.) IN NET-
C              WORK LEVEL J.
C        IDXO: LIST OF 0-COORDINATES IN CSPFE OF THE GATE UNDER
C              CONSIDERATION.
C       IDXOE: LIST OF 0-ERROR-COORDINATES IN CSPFE OF THE GATE UNDER
C              CONSIDERATION.
C        IDX1: LIST OF 1-COORDINATES IN CSPFE OF THE GATE UNDER
C              CONSIDERATION.
C       IDX1E: LIST OF 1-ERROR-COORDINATES IN CSPFE OF THE GATE UNDER
```

```
C             CONSIDERATION.
C       IFLAG: SAME AS EYEFLG IN SUBROUTINE PROCII.
C      INC$MX: INC$MX(GI,GJ)>0 MEANS THERE EXISTS A CONNECTION FROM GATE
C              (OR EX. VAR.) GI TO GATE GJ.  INC$MX(GI,GJ)=0 IF NOT.
C      INPTCV: LISTS FOR EACH CORRESPONDING ENTRY OF F$1, HOW MANY INPUTS
C              HAVE A '1' IN THE POSITION INDICATED BY F$1.
C       IPATH: IPATH(GI)=1 MEANS GATE GI IS ON A PATH FROM A CERTAIN GATE
C              TO AN OUTPUT GATE.  OTHERWISE IPATH(GI) = 0.
C       IPRED: IPRED(I,GJ) GIVES THE NAME OF THE I-TH GATE OR EX. VAR. IN
C              A LIST OF GATES AND EX. VAR. FEEDING GJ.
C       ISUCC: ISUCC(I,GJ) GIVES THE NAME OF THE I-TH GATE FED BY GJ.
C       JFLAG: SAME AS JAYFLG IN SUBROUTINE PROCII.
C        KEYA: A FLAG INDICATING IF ANY ERROR COMPENSATION HAS BEEN
C              PERFORMED.
C        KEYB: A FLAG INDICATING IF ANY PRIMARY 0-ERROR-COORDINATES HAS
C              BEEN COMPENSATED.
C       KFLAG: SAME AS KEIFLG IN PROCII.
C        LEVM: NUMBER OF LEVELS IN THE NETWORK (NOTE EX. VAR. ARE ALSO
C              ASSIGNED LEVELS JUST LIKE GATES).
C      LGLIST: LGLIST(J) TELLS NO. OF GATES AND EX. VAR. IN LEVEL J OF
C              NETWORK.
C         LIP: NUMBER OF PREDECESSORS FOR THE GATE UNDER CONSIDERATION.
C      LIPRED: LIPRED(GI) TELLS NO. OF IMMEDIATE PREDECESSORS OF GATE GI.
C       LISTC: ORDERED LIST OF CONNECTIBLE INPUTS TO GCO.  ORDERED BY
C              DECREASING NO. OF 0'S IN GCO COVERED.
C       LISTL: ORDERED LIST OF GATES AND EX. VAR. WHICH ORIGINALLY FED
C              GCO AND WHICH HAVE NOT YET BEEN DISCONNECTED.  ORDERED BY
C              DECREASING NO. OF ESSENTIAL 1'S.
C      LISUCC: LISUCC(GI) TELLS NO. OF IMMEDIATE SUCCESSORS OF GATE (OR
C              EX. VAR.) GI.
C       LMTS2: UPPER LIMIT OF THE NUMBER OF ELEMENTS IN SET S2.
C      LPOTAB: FOR GATE GI, LPOTAB(GI) POINTS TO LAST ROW OF POTAB
C              CONCERNING GI.
C           M: NUMBER OF NETWORK OUTPUT GATES.
C           N: NUMBER OF EXTERNAL VARIABLES (OR INPUT FNC.) AVAILABLE.
C      NEPMAX: FOR ERROR COMPENSATION PROGRAMS.  IF MORE THAN NEPMAX
C              ERROR POSITIONS OCCUR WHEN A PARTICULAR GATE IS REMOVED,
C              PROGRAM SKIPS ATTEMPT TO COMPENSATE FOR THAT GATE'S
C              REMOVAL.  VALUE CAN BE SPECIFIED BY USER, OTHERWISE EQUAL
C              TO ONE HALF OF N2 BY DEFAULT.
C          NM: SUM OF N PLUS M
C         NM1: SUM OF NM PLUS 1.
C         NN2: PRODUCT OF N AND N2.
C         NOS: NUMBER OF ELEMENTS IN SET S.
C        NOS1: NUMBER OF ELEMENTS IN SET S1.
C      NOS1SV: NUMBER OF ELEMENTS IN SET S1 BEFORE ENTERING SUBROUTINE
C              RPLCF.
C        NOS2: NUMBER OF ELEMENTS IN SET S2.
C        NOT1: NUMBER OF ELEMENTS IN SET T1.
C      NOT1SV: NUMBER OF ELEMENTS IN SET T1 BEFORE ENTERING SUBROUTINE
C              RPLCF.
C         NO0: NUMBER OF ACTIVE ELEMENTS IN ARRAY IDX0.
C        NO0E: NUMBER OF ACTIVE ELEMENTS IN ARRAY IDX0E.
C         NO1: NUMBER OF ACTIVE ELEMENTS IN ARRAY IDX1.
C        NO1E: NUMBER OF ACTIVE ELEMENTS IN ARRAY IDX1E.
C          NR: SUM OF N PLUS R.
C        NRN2: PRODUCT OF NR AND N2.
C       NRPLC: NRPLC(I) STORES THE NUMBER OF ELEMENTS IN RPLC(I,*)
C                                                      FOR I=1,2.
C          N1: SUM OF N PLUS 1.
C          N2: NUMBER OF DIFFERENT INPUT COMBINATIONS TO BE CONSIDERED
```

```
C           (USUALLY 2 TO THE POWER N).
C     ORIGIN: ORIGIN(GI)=1 MEANS GI ORIGINALLY CONNECTED TO GCO.
C             ORIGIN(GI)=0 MEANS GI DID NOT FEED GCO ORIGINALLY.
C         P$: P$(1,-) CONSECUTIVELY LISTS OUTPUTS OF EVERY EX. VAR. AND
C             EVERY GATE (FOR EVERY INPUT COMBINATION): P$(1,1),...,
C             P$(1,N2) FOR FIRST EX VAR; P$(1,N2+1),...,P$(1,2*N2) FOR
C             SECOND EX VAR; ... ; P$(1,N*N2+1),..., P$(1,N*N2+N2) FOR
C             FIRST GATE; ETC.  P$(2,-) IS USED AS WORK SPACE FOR
C             CALCULATIONS ASSOCIATED WITH P$(1,-).
C        PCO: FOR ERROR COMPENSATION PROCEDURES.  PCO IS THE GATE
C             REMOVED FROM ORIGINAL NETWORK TO OBTAIN CURRENT ALTERED
C             NETWORK.
C     POINTA: NOT USED.
C     POINTC: POINTS TO LAST ELEMENT IN LISTC.
C     POINTL: POINTS TO LAST ELEMENT IN LISTL.
C     POINTR: POINTS TO LAST ELEMENT IN RNEC1 (IN SUBROUTINE SUBSTI).
C      POTAB: POSSIBLE OUTPUT TABLE.  HOLDS INFORMATION ABOUT ALL
C             COMBINATIONS OF CONNECTIONS TO FORM NEW (AND HOPEFULLY
C             USEFUL) FUNCTIONS.
C     PPOTAB: FOR GATE GI, PPOTAB(GI) POINTS TO FIRST OF A SEQUENCE OF
C             ROWS OF POTAB CONCERNING GI.
C          R: NUMBER OF GATES IN THE NETWORK (EXCLUDES EX VAR, ALSO
C             NOTE SOME OF R GATES MAY BE ISOLATED).
C       RPLC: RPLC(1,*) STORES THE SELECTED GATE'S IP GATES WHICH HAVE
C                       ERROR-COORDINATES OF WEIGHT 2 OR ABOVE.
C             RPLC(2,*) STORES THE SELECTED GATE'S IP GATES WHICH HAVE
C                       AT LEAST ONE ERROR-COORDINATE OF WEIGHT 1.
C     RSCONN: LIST OF CONNECTIONS ADDED TO A NETWORK (IN CODED FORM).
C     RTCONN: LIST OF CONNECTIONS REMOVED FROM A NETWORK (CODED FORM).
C          S: NO. OF CONNECTIONS ADDED TO A NETWORK.  POINTS TO LAST
C             ENTRY IN RSCONN.
C       SETS: SET S CONSISTING OF INPUTS OF THE GATE UNDER CONSIDERATION
C             WHICH ARE TO BE REPLACED IF POSSIBLE.
C      SETS1: SET S1 CONSISTING OF ELEMENTS OF SET S WHICH CAN BE
C             REPLACED BY ELEMENTS IN SET S2.
C      SETS2: SET S2 CONSISTING OF FUNCTIONS WHICH ARE CANDIDATES FOR
C             REPLACING ELEMENTS IN SET S.
C      SETT1: SET T1 CONSISTING OF ESSENTIAL ONES COVERED BY ELEMENTS IN
C                                                             SET S1.
C        STS: STARTING ELEMENT OF SET S.
C     SUC$MX: SUC$MX(GI,GJ)>0 MEANS GATE GJ IS A SUCCESSOR OF GATE GI.
C             SUC$MX(GI,GJ)=0 IF NOT.
C       SUMP: SUM OF ALL ACTIVE INPUTS OF THE GATE UNDER CONSIDERATION.
C      SUMS2: SUM OF ALL ACTIVE ELEMENTS OF SET S2.
C          T: NUMBER OF CONNECTIONS REMOVED FROM A NETWORK.  POINTS TO
C             LAST ENTRY IN RTCONN.
C       TIME: USED TO STORE AMOUNT OF ELAPSED COMPUTATION TIME.
C      UNAME: MNEMONIC NAMES FOR EXTERNAL VARIABLES AND GATES.
C    VF$UB1: POINTS TO LAST ELEMENT IN VF$1.
C       VF$1: SIMILAR TO F$1, EXCEPT THIS LISTS JUST COMPONENT POSITIONS
C             (OF 0'S IN CSPF VECTOR OF GCO) COVERED ONLY BY REMAINING
C             ORIGINALLY CONNECTED INPUTS TO GCO.
C
C
C
      COMMON NEPMAX
      COMMON    N           , M            , A            , B
     1     ,    R           , N2           , N1           , NR
     2     ,    NM          , KFLAG        , JFLAG        , COST
     3     ,    LEVM        , NRN2         , NM1          , NN2
      COMMON    ISUCC(40,40) , LISUCC(40)   , IPRED(40,40) , LIPRED(40)
```

```
     1    ,   INC$MX(40,40), SUC$MX(40,40), P$(2,1280)    , UNAME(40)
     2    ,   GLEVEL(40)    , LGLIST(40)    , HLIST(40,40) , TIME
      COMMON   T            , RTCONN(100)   , S            , RSCONN(100)
      COMMON   IFLAG        ,POINTA         ,ESS1S(40)     ,F$1(32)
     1     ,F$UB1           ,INPTCV(32)     ,LISTC(40)     ,POINTC
     2     ,LISTL(40)       ,POINTL         ,ORIGIN(40)    ,IPATH(40)
     3     ,POINTR          ,VF$1(32)       ,VF$UB1        ,GSMALL(40,32)
      COMMON   POTAB(200,42),PPOTAB(40)     ,LPOTAB(40)    ,NRPLC(2)
     1     ,RPLC(2,40)      ,IDX0(32)       ,IDX0E(32)     ,IDX1(32)
     2     ,IDX1E(32)       ,SUMP(32)       ,SETT1(32)     ,NOT1
     3     ,SETS1(40)       ,NOS1           ,SETS(40)      ,NOS
     4     ,STS             ,SUMS2(32)      ,SETS2(200)    ,NOS2
     5     ,LIP             ,NO0E           ,KEYA          ,KEYB
     6     ,NO0             ,NO1            ,NO1E          ,$GT
     7     ,$LTH            ,$PW            ,$NOE          ,GI
      COMMON                NOT1SV        ,NOS1SV        ,LMTS2
      DIMENSION CNTLIS(144),UGATE(40),UHEAD(20)
      DATA KOUNT5 /0/, UBLANK/'    '/
  990 READ(5,1000,END=500) UHEAD, N, M, R, A, B, UC, NEPMAX
C     NEPMAX IS THE MAXIMUM ALLOWABLE NUMBER OF ERROR POSITIONS
 1000 FORMAT(20A4/5I4,A4,I4)
      KEYXC=0
      IF(UC.NE.UBLANK) KEYXC=1
      CALL PAGE
      CALL LINE(10)
      KOUNT5=KOUNT5+1
      PRINT 2, KOUNT5
    2 FORMAT(20X,'*** OPTIMAL NOR NETWORK ***',50X,'PROBLEM NO.= ',I4 )
      CALL LINE(4)
      PRINT 1005, UHEAD
 1005 FORMAT(25X,20A4)
      CALL LINE(4)
      PRINT 10, N,M,A,B
   10 FORMAT(30X,'NUMBER OF VARIABLES =',I4 //
     1        30X,'NUMBER OF FUNCTIONS =',I4 //
     2        30X,'COST COEFFICIENT A  =',I4//
     3        47X,                'B   =',I4)
      CALL LINE(1)
      IF(KEYXC.NE.0) GO TO 25
      PRINT 21
   21 FORMAT(1H0,29X,'--- UNCOMPLEMENTED VARIABLES  X ---')
      GO TO 30
   25 CONTINUE
      PRINT 28
   28 FORMAT(1H0,29X,'--- BOTH COMPLEMENTED AND UNCOMPLEMENTED VARIABLES
     1 X, Y ---')
   30 CONTINUE
      CALL LINE(5)
C***** SET UP EXTERNAL VARIABLES *****
      N2=2**N
      IF(NEPMAX.EQ.0)NEPMAX = N2/2
      H=N*N2
      J=N2
      L= 1
      I=0
      DO 1011 II=1,N
       J=J/2
       L=L*2
       SN= 1
       DO 1010 LL=1,L
        SN=-SN
```

```
      V=(1+SN)/2
      DO 1009 JJ=1,J
       I=I+1
       P$(1,I)=V
      IF(KEYXC.NE.0)P$(1,I+H)=1-V
 1009   CONTINUE
 1010  CONTINUE
 1011 CONTINUE
      IF(KEYXC.NE.0) N=N+N
      N1=N+1
      NM=N+M
      NM1=NM+1
      NN2=N*N2+1
      NR=N+R
      NRN2=NR*N2
      CALL OUTPUT(INC$MX,KEYXC)
C***** READ IN NETWORK INFORMATION AND SET UP INC$MX *****
      READ 1001,   CNTLIS
 1001 FORMAT(16I5)
      DO 1115 GI=1,NR
      DO 1115 GJ=1,NR
 1115 INC$MX(GI,GJ)=0
      DO 1120 I=1,144
       ITEM=CNTLIS(I)
      IF(ITEM.EQ.0) GO TO 1119
       GI=ITEM/100
       GJ=ITEM-100*GI
       INC$MX(GI,GJ)=1
       GO TO 1120
 1119 COST=A*R+B*(I-1)
       GO TO 1130
 1120 CONTINUE
 1130 CONTINUE
      CALL SUBNET
      CALL PVALUE
      CALL LINE(4)
      PRINT 1140, COST
 1140 FORMAT(20X,' ORIGINAL NETWORK    COST=', I5)
      CALL LINE(4)
      CALL TRUTH(P$,1)
      CALL LINE(4)
      CALL CKT(INC$MX,GLEVEL)
C
C***** ENTRY REDUNDANCY CHECK *****
      S = 0
      T = 0
      CALL UNNECE
      GATES = M
      C = 0
      DO 4 GI =  1,NR
      C = C + LISUCC(GI)
      IF(GI.LE.NM)GOTO4
      IF(LISUCC(GI).GT.0)GATES=GATES+1
    4 CONTINUE
      OLDCST = A*GATES + B*(C)
      T=0
      S=0
C     INITIALIZE TIMER TO 10 MINUTES
      CALL STIMEZ(60000)
      TIME = KTIMEZ(0)
C****   PROCEDURE  PROCV
```

```
      CALL PROCV
C     CALL FOR ELAPSED TIME
      TIME = KTIMEZ(0) - TIME
      CALL LINE(4)
      PRINT 3915
 3916 FORMAT(20X,'TIME ELAPSED =',I8,'  CENTISECONDS')
 3915 FORMAT(20X,'NETWORK DERIVED BY PROCV ')
      PRINT 3916,TIME
      CALL LINE(4)
      CALL TRUTH(P$,1)
      CALL LINE(4)
      CALL CKT(INC$MX,GLEVEL)
      GATES = M
      C = 0
      DO 36 GI = 1,NR
      C = C + LISUCC(GI)
      IF(GI.LE.NM) GO TO 36
      IF(LISUCC(GI).GT.0) GATES = GATES + 1
   36 CONTINUE
      NEWCST = A*GATES + B*C
      IF(NEWCST.LT.OLDCST)GO TO 37
      PRINT 105
  105 FORMAT(1H ,10X,'NO REDUNDANCY FOUND.')
      GO TO 990
   37 CALL LINE(3)
      PRINT 320,NEWCST
  320 FORMAT(9X,'* A NETWORK DERIVED BY PROCV '/9X,' COST=',I5,'.')
      GO TO 990
  500 STOP
      END


      SUBROUTINE OUTPUT(MATRIX,ARRAY)
      IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)
C
C     DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.
C
      COMMON NEPMAX
      COMMON    N               , M               , A               , B
     1     ,    R               , N2              , N1              , NR
     2     ,    NM              , KFLAG           , JFLAG           , COST
     3     ,    LEVM            , NRN2            , NM1             , NN2
      COMMON    ISUCC(40,40)    , LISUCC(40)      , IPRED(40,40)    , LIPRED(40)
     1     ,    INC$MX(40,40)   , SUC$MX(40,40)   , P$(2,1280)      , UNAME(40)
     2     ,    GLEVEL(40)      , LGLIST(40)      , HLIST(40,40)    , TIME
      COMMON    T               , RTCONN(100)     , S               , RSCONN(100)
      COMMON    IFLAG           ,POINTA           ,ESS1S(40)        ,F$1(32)
     1     ,F$UB1               ,INPTCV(32)       ,LISTC(40)        ,POINTC
     2     ,LISTL(40)           ,POINTL           ,ORIGIN(40)       ,IPATH(40)
     3     ,POINTR              ,VF$1(32)         ,VF$UB1           ,GSMALL(40,32)
      COMMON    POTAB(200,42)   ,PPOTAB(40)       ,LPOTAB(40)       ,NRPLC(2)
     1     ,RPLC(2,40)          ,IDX0(32)         ,IDX0E(32)        ,IDX1(32)
     2     ,IDX1E(32)           ,SUMP(32)         ,SETT1(32)        ,NCT1
     3     ,SETS1(40)           ,NOS1             ,SETS(40)         ,NOS
     4     ,STS                 ,SUMS2(32)        ,SETS2(200)       ,NOS2
     5     ,LIP                 ,NOOE             ,KEYA             ,KEYB
     6     ,NOO                 ,NO1              ,NO1E             ,$GT
     7     ,$LTH                ,$PW              ,$NOE             ,GI
      COMMON                    NOT1SV            ,NOS1SV           ,LMTS2
      DIMENSION UX(5), UY(5), UG(40), UF(40), ARRAY(40), ARRAY2(2,1280)
      DIMENSION MATRIX(40,40)
```

```
      DATA UX /' X1',' X2',' X3',' X4',' X5'/
      DATA UY/' Y1',' Y2',' Y3',' Y4',' Y5'/
      DATA UF /'  1','  2','  3','  4','  5','  6','  7','  8'
     1        , '  9',' 10',' 11',' 12',' 13',' 14',' 15',' 16'
     2        , ' 17',' 18',' 19',' 20',' 21',' 22',' 23',' 24'
     3        , ' 25',' 26',' 27',' 28',' 29',' 30',' 31',' 32'
     4        , ' 33',' 34',' 35',' 36',' 37',' 38',' 39',' 40'/
      DATA GMAX/40/
C
      KEYXC=ARRAY(1)
      IF(KEYXC.NE.0) GO TO 50
      DO 1 GI=1,N
       UNAME(GI)=UX(GI)
    1 CONTINUE
      GO TO 100
   50 CONTINUE
      L=N/2
      DO 4 GI=1,L
       UNAME(GI)=UX(GI)
      UNAME(GI+L)=UY(GI)
    4 CONTINUE
  100 CONTINUE
      DO 2 GI=N1,GMAX
       UNAME(GI)=UF(GI-N)
    2 CONTINUE
      RETURN
C
      ENTRY LINE(L)
      DO 6 LL=1,L
       PRINT 5
    5 FORMAT(1H )
    6 CONTINUE
      RETURN
C
      ENTRY PAGE
      PRINT 7
    7 FORMAT(1H1)
      RETURN
C
      ENTRY CKT(MATRIX,ARRAY)
      PRINT 10
   10 FORMAT(1H ,8X,'GATE .. LEVEL',6X, 'FED BY'/)
      DO 20 GJ=N1,NR
       G=0
       DO 15 GI=1,NR
        IF(MATRIX(GI,GJ).EQ.0) GO TO 15
        G=G+1
        UG(G)=UNAME(GI)
   15  CONTINUE
       IF(G.EQ.0) GO TO 18
       PRINT 17, UNAME(GJ),ARRAY(GJ),(UG(GG),GG=1,G)
   17  FORMAT(1H0, 9X,A3,5X,'/',I2,'/',5X,35(    A3))
       GO TO 20
   18  PRINT 19, UNAME(GJ),ARRAY(GJ)
   19  FORMAT(1H0, 9X,A3,5X,'/',I2,'/')
   20 CONTINUE
      RETURN
C
      ENTRY TRUTH(ARRAY2,J)
      IF(J.EQ.2) GO TO 36
      PRINT 35
```

```
   35 FORMAT(11X,    'TRUTH TABLE'/)
      GO TO 38
   36 PRINT 37
   37 FORMAT(11X,   'REQUIREMENT TABLE')
   38 CONTINUE
      DO 40 GI=1,NR
       ILO=(GI-1)*N2+1
       IHI=ILO+N2-1
       PRINT 41, UNAME(GI), (ARRAY2(J,I),I=ILO,IHI)
   40 CONTINUE
   41 FORMAT(1HO, 9X,A3,' = ', 32(I1,1X))
      RETURN
      END


      SUBROUTINE PROCV
C**** THIS SUBROUTINE HAS AN ENTRY POINT PROCVS(GI) BESIDE PROCV ITSELF
C     WHEN THIS SUBROUTINE IS ENTERED FROM PROCV, PROCEDURE V IS APPLIED
C     TO ALL GATES IN THE NETWORK REPEATEDLY UNTIL NO GATE CAN BE
C     REMOVED BY THIS PROCEDURE. WHEN THIS SUBROUTINE IS ENTERED AT
C     ENTRY POINT PROCVS(GI), ONLY GATE GI IS EXAMINED TO SEE WHETHER IT
C     CAN BE REMOVED BY PROCEDURE V OR NOT *******
      IMPLICIT INTEGER*4(A-T,V-Z), REAL(U)
C
C     DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.
C
      COMMON NEPMAX
      COMMON    N              , M              , A              , B
     1     ,    R              , N2             , N1             , NR
     2     ,    NM             , KFLAG          , JFLAG          , COST
     3     ,    LEVM           , NRN2           , NM1            , NN2
      COMMON    ISUCC(40,40)   , LISUCC(40)     , IPRED(40,40)   , LIPRED(40)
     1     ,    INC$MX(40,40)  , SUC$MX(40,40)  , P$(2,1280)     , UNAME(40)
     2     ,    GLEVEL(40)     , LGLIST(40)     , HLIST(40,40)   , TIME
      COMMON    T              , RTCONN(100)    , S              , RSCONN(100)
      COMMON    IFLAG          ,POINTA          ,ESS1S(40)       ,F$1(32)
     1      ,F$UB1             ,INPTCV(32)      ,LISTC(40)       ,POINTC
     2      ,LISTL(40)         ,POINTL          ,ORIGIN(40)      ,IPATH(40)
     3      ,POINTR            ,VF$1(32)        ,VF$UB1          ,GSMALL(40,32)
      COMMON    POTAB(200,42),PPOTAB(40)        ,LPOTAB(40)      ,NRPLC(2)
     1      ,RPLC(2,40)        ,IDX0(32)        ,IDX0E(32)       ,IDX1(32)
     2      ,IDX1E(32)         ,SUMP(32)        ,SETT1(32)       ,NOT1
     3      ,SETS1(40)         ,NOS1            ,SETS(40)        ,NOS
     4      ,STS               ,SUMS2(32)       ,SETS2(200)      ,NOS2
     5      ,LIP               ,NOOE            ,KEYA            ,KEYB
     6      ,NOO               ,NO1             ,NO1E            ,$GT
     7      ,$LTH              ,$PW             ,$NOE            ,GI$$$$
      COMMON                   NOT1SV         ,NOS1SV          ,LMTS2
      COMMON    /VRQRNW/         GIGJ           , PGIGJ(1280)    , LISTIJ(40)
      DIMENSION INDEXO(32), INDEX1(32), LISCND(40), ISCND(40,40),
     1          CNCTBL(40), NCTBL(40,40), TEMPRS(90), INDEXS(32)
      ETRKEY = 0
C***** SELECT GI WHICH IS GOING TO BE REMOVED *****
    1 TS = T
      GI=NM
    2 GI = GI + 1
      IF(GI.LE.NR) GO TO 3
      IF(T.EQ.TS) RETURN
      GO TO 1
C**** ACTUAL ENTRY POINT FOR PROCVS(GI)  ********
    3 CONTINUE
```

```
      IF(GLEVEL(GI).EQ.1)GO TO 199
C***** SKIP GATES WHICH HAVE ONLY ONE OUTPUT TO GI *****
      LIP=LIPRED(GI)
      DO 10 LI=1,LIP
       GJ=IPRED(LI,GI)
       IF(GJ.LE.N)GO TO 10
       IF(LISUCC(GJ).EQ.1.AND.ISUCC(1,GJ).GT.NM)GO TO 199
   10 CONTINUE
C
C***** GATES WHICH HAVE ONLY ONE INPUT FROM GI ARE REDUNDANT *****
      LIS=LISUCC(GI)
      DO 11 LI=1,LIS
       GJ=ISUCC(LI,GI)
       IF(LISUCC(GJ).EQ.0)GO TO 11
       IF(GJ.GT.NM.AND.LIPRED(GJ).EQ.1) GO TO 12
   11 CONTINUE
      GO TO 20
C
C***** GJ IS REDUNDANT. DISCONNECT CONNECTION GI TO GJ AND CONNECT ALL
C      INPUTS OF GI TO EVERY SUCCEEDING GATE OF GJ *****
   12 LIS=LISUCC(GJ)
      T=T+1
      RTCONN(T)=100*GI+GJ
      INC$MX(GI,GJ)=0
      DO 15 LI=1,LIS
       GK=ISUCC(LI,GJ)
       T=T+1
       RTCONN(T)=100*GJ+GK
       INC$MX(GJ,GK)=0
       DO 13 IP=1,LIP
        GH=IPRED(IP,GI)
        IF(INC$MX(GH,GK).GE.1)GO TO 13
        S=S+1
        RSCONN(S)=100*GH+GK
        INC$MX(GH,GK)=1
   13  CONTINUE
   15 CONTINUE
      CALL SUBNET
     CALL UNNECE
      GO TO 199
C
C***** CALL RQRNW TO CALCULATE REQUIREMENTS FOR EACH GATE *****
   20 TEMPS=0
      CALL RQRNW(GI)
C
C***** SELECT GATES TO REPLACE CONNECTIONS FROM GI *****
      DO 169 GIG=1,GIGJ
       GJ=LISTIJ(GIG)
       BSGIGJ=(GIG-1)*N2
C***** LIST "0" AND "1" COMPONENTS IN REQUIRED FUNCTION OF GI TO GJ ****
       NO1=0
       NO0=0
       DO 52 TH=1,N2
        IF(PGIGJ(BSGIGJ+TH))52,50,51
   50   NO0=NO0+1
        INDEXO(NO0)=TH
        GO TO 52
   51   NO1=NO1+1
        INDEX1(NO1)=TH
   52  CONTINUE
       IF(NO1.EQ.0) GO TO 169
```

```
      NO11=NO1
C
C***** PARTITION GATES NOT FED BY GI INTO THREE CATAGORIES:
C      1.NCTBL: CONNECTIBLE GATES
C      2.ISCND: MAY BE MADE CONNECTIBLE GATES
C      3.OTHERS *****
        CNDKNT=0
        CBLKNT=0
       DO 56 GK=1,NR
        IF(GK.EQ.GI) GO TO 56
        IF(GK.GT.NM.AND.GLEVEL(GK).EQ.1) GO TO 56
        IF(SUC$MX(GI,GK).GE.1)GO TO 56
        BSGK=(GK-1)*N2
C
C***** COMPARE GK WITH PGIGJ *****
        EFSIGN=0
C***** EFSIGN=1: EFFECTIVE; EFSIGN=0: NON-EFFECTIVE *****
        DO 53 NRUN=1,NO1
         IF(P$(1,BSGK+INDEX1(NRUN)).EQ.0)GO TO 53
         EFSIGN=1
   53   CONTINUE
        IF(EFSIGN.EQ.0)GO TO 56
C
        EFSIGN=0
C***** EFSIGN=0: CONNECTABLE; EFSIGN=1: CANDIDATE *****
        DO 54 NRUN=1,NO0
         IF(P$(1,BSGK+INDEXO(NRUN)).EQ.0)GO TO 54
         IF(P$(2,BSGK+INDEXO(NRUN)).EQ.1)GO TO 56
         EFSIGN=1
   54   CONTINUE
        IF(EFSIGN.EQ.0)GO TO 55
        IF(GK.LE.NM)GO TO 56
        CNDKNT=CNDKNT+1
        ISCND(CNDKNT,GIG)=GK
        GO TO 56
   55   CBLKNT=CBLKNT+1
        NCTBL(CBLKNT,GIG)=GK
   56  CONTINUE
       LISCND(GIG)=CNDKNT
       CNCTBL(GIG)=CBLKNT
C
C***** SELECT ESSENTIAL GATES FROM NCTBL *****
       CNCT=CNCTBL(GIG)
       IF(CNCT.EQ.0)GO TO 62
       DO 59 CNC=1,CNCT
        EFSIGN=0
        GK=NCTBL(CNC,GIG)
        BSGK=(GK-1)*N2
        DO 57 NRUN=1,NO1
         IF(INDEX1(NRUN).GT.100)GO TO 57
         TH=INDEX1(NRUN)
         IF(P$(1,BSGK+TH).NE.1)GO TO 57
         INDEX1(NRUN)=100+TH
         P$(2,BSGK+TH)=1
         EFSIGN=1
   57   CONTINUE
        IF(EFSIGN.EQ.0)GO TO 59
        TEMPS=TEMPS+1
        TEMPRS(TEMPS)=100*GK+GJ
        INC$MX(GK,GJ)=1
        CALL SUBNET
```

```
C
C***** UPDATE REQUIREMENTS FOR GK *****
      IF(GK.LE.NM)GO TO 59
      DO 58 NRUN=1,NOO
   58 P$(2,BSGK+INDEXO(NRUN))=0
      CALL RQRGT(GK)
   59 CONTINUE
C
C***** COMPRESS THE TABLE OF UNCOVERED ONE COMPONENTS *****
      NO11=0
      DO 61 NRUN=1,NO1
       IF(INDEX1(NRUN).GT.100)GO TO 61
       NO11=NO11+1
       INDEX1(NO11)=INDEX1(NRUN)
   61 CONTINUE
C
C***** GATES CONNECTED TO GJ CAN REPLACE CONNECTION GI TO GJ(NO11=0)****
      IF(NO11.EQ.0)GO TO 169
C
C***** FIND CANDIDATES FOR EACH UNCOVERED ONE *****
C
   62 LISCN=LISCND(GIG)
      IF(LISCN.EQ.0)GO TO 185
      DO 168 LISC=1,LISCN
       GK=ISCND(LISC,GIG)
       BSGK=(GK-1)*N2
       DO 63 NRUN=1,NO11
        TTH=INDEX1(NRUN)
        IF(P$(1,BSGK+TTH).EQ.1)GO TO 64
   63  CONTINUE
       GO TO 168
C
C***** CHECK WHICH COMPONENT OF GK HAS TO BE CHANGED *****
C
   64  NO10=0
       DO 65 NRUN=1,NOO
        IF(P$(1,BSGK+INDEXO(NRUN)).NE.1)GO TO 65
        IF(P$(2,BSGK+INDEXO(NRUN)).GE.0)GO TO 168
        NO10=NO10+1
        INDEXS(NO10)=INDEXO(NRUN)
   65  CONTINUE
C
C***** FIND A FUNCTION GL, CONNECTING WHICH TO GK WILL MAKE GK
C                                             CONNECTIBLE TO GJ *****
       DO 90 GL=1,NR
        IF(GL.GT.NM.AND.GLEVEL(GL).EQ.1) GO TO 90
        IF(SUC$MX(GI,GL).GE.1.OR.SUC$MX(GK,GL).GE.1) GO TO 90
C
C***** COMPARE FUNCTION GL WITH REQUIREMENTS FOR GJ *****
        BSGL=(GL-1)*N2
        IF(P$(1,BSGL+TTH).NE.0)GO TO 90
        DO 67 NRUN=1,NO10
         TH=INDEXS(NRUN)
         IF(P$(1,BSGL+ TH).NE.1)GO TO 90
   67   CONTINUE
        DO 69 TH=1,N2
         IF(P$(2,BSGK+TH))69,69,68
   68    IF(P$(1,BSGL+TH).EQ.1)GO TO 90
   69   CONTINUE
C
C***** CONNECT GL TO GK *****
```

```
C
         TEMPS=TEMPS+1
         TEMPRS(TEMPS)=100*GL+GK
         INC$MX(GL,GK)=1
         TEMPS=TEMPS+1
         TEMPRS(TEMPS)=100*GK+GJ
         INC$MX(GK,GJ)=1
         CALL SUBNET
C
C***** UPDATE TRUTH TABLE FOR GK *****
         DO 71 TH=1,N2
          IF(P$(1,BSGL+TH))71,71,70
   70     P$(1,BSGK+TH)=0
   71    CONTINUE
      CALL UPTRTH(GI, GK)
C
C***** UPDATE REQUIREMENT TABLE FOR GK *****
         DO 72 NRUN=1,NCO
          P$(2,BSGK+INDEXO(NRUN))=0
   72    CONTINUE
         NO1=NO11
         NO11=0
         DO 74 NRUN=1,NO1
          TH=INDEX1(NRUN)
          IF(P$(1,BSGK+TH).EQ.1)GO TO 73
          NO11=NO11+1
          INDEX1(NO11)=TH
          GO TO 74
   73     P$(2,BSGK+TH)=1
   74    CONTINUE
         CALL RQRGT(GK)
         IF(NO11.EQ.0)GO TO 169
      GO TO 168
   90   CONTINUE
  168  CONTINUE
       GO TO 185
  169  CONTINUE
C
C***** GI CAN BE ELIMINATED *****
C***** LIST ALL ADDED CONNECTIONS *****
      IF(TEMPS.LE.0) GO TO 173
       DO 172 TEM=1,TEMPS
        S=S+1
        RSCONN(S)=TEMPRS(TEM)
  172  CONTINUE
C
C***** DISCONNECT ALL CONNECTIONS RELATED TO GI *****
  173 CONTINUE
       LIS=LISUCC(GI)
       DO 175 LI=1,LIS
        GJ=ISUCC(LI,GI)
        T=T+1
        RTCONN(T)=100*GI+GJ
        INC$MX(GI,GJ)=0
  175  CONTINUE
       LIP=LIPRED(GI)
       DO 176 LI=1,LIP
        GJ=IPRED(LI,GI)
        T=T+1
        RTCONN(T)=100*GJ+GI
        INC$MX(GJ,GI)=0
```

```
176  CONTINUE
     CALL SUBNET
    CALL UNNECE
     CALL PVALUE
     GO TO 199
C
C***** GI CAN NOT BE ELIMINATED. RESTORE THE ORINGINAL NETWORK *****
 185  IF(TEMPS.EQ.0)GO TO 199
      DO 189 TEM=1,TEMPS
       GL=TEMPRS(TEM)/100
       GK=TEMPRS(TEM)-GL*100
     IF(GL.GT.N.OR.GK.GT.NM) GO TO 188
     S=S+1
     RSCONN(S)=TEMPRS(TEM)
     GO TO 189
 188 CONTINUE
      INC$MX(GL,GK)=0
 189  CONTINUE
      CALL SUBNET
      CALL PVALUE
 199 CONTINUE
     IF(ETRKEY.EQ.0) GO TO 2
     RETURN
     ENTRY PROCVS(GI)
     ETRKEY=1
     GO TO 3
     END


     SUBROUTINE RQRNW(GI)
     IMPLICIT INTEGER*4(A-T,V-Z), REAL(U)
C
C    DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.
C
     COMMON NEPMAX
     COMMON    N              , M              , A              , B
    1     ,    R              , N2             , N1             , NR
    2     ,    NM             , KFLAG          , JFLAG          , COST
    3     ,    LEVM           , NRN2           , NM1            , NN2
     COMMON    ISUCC(40,40)   , LISUCC(40)     , IPRED(40,40)   , LIPRED(40)
    1     ,    INC$MX(40,40)  , SUC$MX(40,40)  , P$(2,1280)     , UNAME(40)
    2     ,    GLEVEL(40)     , LGLIST(40)     , HLIST(40,40)   , TIME
     COMMON    T              , RTCONN(100)    , S              , RSCONN(100)
     COMMON    IFLAG          ,POINTA          ,ESS1S(40)       ,F$1(32)
    1     ,F$UB1              ,INPTCV(32)      ,LISTC(40)       ,POINTC
    2     ,LISTL(40)          ,POINTL          ,ORIGIN(40)      ,IPATH(40)
    3     ,POINTR             ,VF$1(32)        ,VF$UB1          ,GSMALL(40,32)
     COMMON    PDTAB(200,42),PPDTAB(40)        ,LPDTAB(40)      ,NRPLC(2)
    1     ,RPLC(2,40)         ,IDX0(32)        ,IDX0E(32)       ,IDX1(32)
    2     ,IDX1E(32)          ,SUMP(32)        ,SETT1(32)       ,NOT1
    3     ,SETS1(40)          ,NOS1            ,SETS(40)        ,NOS
    4     ,STS                ,SUMS2(32)       ,SETS2(200)      ,NOS2
    5     ,LIP                ,NOOE            ,KEYA            ,KEYB
    6     ,NO0                ,NO1             ,NO1E            ,$GT
    7     ,$LTH               ,$PW             ,$NOE            ,GI$$$$
     COMMON                    NOT1SV         ,NOS1SV          ,LMTS2
     COMMON    /VRQRNW/        GIGJ           , PGIGJ(1280)    , LISTIJ(40)
     DIMENSION INDEX0(32), INDEX1(32), PSUM(32)
C
C***** COPY TRUTH TABLE FOR OUTPUT GATES AND EXTERNAL VARIABLES *****
     MTH=NM*N2
```

```
      DO 10 TH=1,MTH
   10 P$(2,TH)=P$(1,TH)
C
C***** INITIALIZE ALL FUNCTIONS TO BE UNDEFINED *****
      MTH1=MTH+1
      MTH2=NR*N2
      DO 11 TH=MTH1,MTH2
   11 P$(2,TH)=-1
C
C***** REQUIREMENTS FOR CONNECTIONS TO A CERTAIN LEVEL *****
      GIGJ=0
      LEVR=LEVM-2
      DO 49 LEV=1,LEVR
       LGL=LGLIST(LEV)
       DO 45 LG=1,LGL
        GJ=HLIST(LG,LEV)
        IF(GJ.EQ.GI)GO TO 45
        IF(GJ.GT.NM.AND.LEV.EQ.1) GO TO 45
        IF(GJ.LE.N) GO TO 45
        BSGJ=(GJ-1)*N2
        LIP=LIPRED(GJ)
        KEYGI=0
        DO 25 TH=1,N2
   25   PSUM(TH)=0
        DO 35 LP=1,LIP
         GK=IPRED(LP,GJ)
         BSGK=(GK-1)*N2
         IF(GK.NE.GI) GO TO 26
         KEYGI=1
         GO TO 35
   26    IF(GK.GT.NM) GO TO 29
C
C***** GK IS EXTERNAL VARIABLE OR OUTPUT GATE, UPDATE PSUM ONLY *****
         DO 28 TH=1,N2
          IF(P$(2,BSGK+TH).EQ.1)PSUM(TH)=PSUM(TH)+1
   28    CONTINUE
         GO TO 35
C***** REQUIREMENTS FOR CONNECTION GK TO GJ *****
   29    DO 33 TH=1,N2
          IF(P$(2,BSGJ+TH)) 33,30,31
   30     IF(PSUM(TH).GE.1.OR.P$(1,BSGK+TH).NE.1) GO TO 33
          P$(2,BSGK+TH)=1
          PSUM(TH)=1
          GO TO 33
   31     P$(2,BSGK+TH)=0
   33    CONTINUE
   35   CONTINUE
        IF(KEYGI.NE.1) GO TO 45
C
C***** CALCULATE REQUIREMENTS FOR CONNECTION GI TO GJ *****
        GIGJ=GIGJ+1
        BSGIGJ=(GIGJ-1)*N2
        LISTIJ(GIGJ)=GJ
        DO 39 TH=1,N2
         IF(P$(2,BSGJ+TH)) 38,36,37
   36    IF(PSUM(TH).GE.1) GO TO 38
         PGIGJ(BSGIGJ+TH)=1
         GO TO 39
   37    PGIGJ(BSGIGJ+TH)=0
         GO TO 39
   38    PGIGJ(BSGIGJ+TH)=-1
```

```
   39   CONTINUE
   45  CONTINUE
   49 CONTINUE
      RETURN
C
C***** UPDATE REQUIREMENTS FOR GATES FEEDING GZ *****
      ENTRY RQRGT(GZ)
      LEV1=GLEVEL(GZ)
      LEV2=LEVM-2
      IF(LEV2.LT.LEV1)RETURN
      DO 189 LEV=LEV1,LEV2
       KEYCHG=0
       LGL=LGLIST(LEV)
       DO 179 LG=1,LGL
        GY=HLIST(LG,LEV)
        IF(GY.LE.N) GO TO 179
        IF(SUC$MX(GY,GZ).LE.0.AND.GY.NE.GZ)GO TO 179
        BSGY=(GY-1)*N2
        LIP=LIPRED(GY)
        NO0=0
        NO1=0
        DO 109 TH=1,N2
         IF(P$(2,BSGY+TH)) 109,107,108
  107    NO0=NO0+1
         INDEXO(NO0)=TH
         GO TO 109
  108    NO1=NO1+1
         INDEX1(NO1)=TH
  109   CONTINUE
        DO 110 TH=1,NO0
  110   PSUM(TH)=0
        DO 119 LI=1,LIP
         GX=IPRED(LI,GY)
C
         BSGX=(GX-1)*N2
         DO 112 NRUN=1,NO0
          IF(P$(2,BSGX+INDEXO(NRUN)))112,112,111
  111     PSUM(NRUN)=PSUM(NRUN)+1
  112    CONTINUE
  119   CONTINUE
        NO00=0
        NO11=0
        DO 125 NRUN=1,NO0
         IF(PSUM(NRUN).GE.1) GO TO 125
         NO00=NO00+1
         INDEXO(NO00)=INDEXO(NRUN)
  125   CONTINUE
        DO 133 LI=1,LIP
         GX=IPRED(LI,GY)
         IF(GX.LE.NM) GO TO 133
         BSGX=(GX-1)*N2
         IF(NO00.EQ.0) GO TO 131
         DO 130 NRUN=1,NO00
          TH=INDEXO(NRUN)
          IF(TH.GT.100) GO TO 130
          IF(P$(1,BSGX+TH).NE.1) GO TO 130
          P$(2,BSGX+TH)=1
          INDEXO(NRUN)=100+TH
  130    CONTINUE
  131    DO 132 NRUN=1,NO1
          TH=INDEX1(NRUN)
```

```
        IF(P$(2,BSGX+TH).NE.-1) GO TO 132
        P$(2,BSGX+TH)=0
        N011=N011+1
  132   CONTINUE
  133   CONTINUE
       IF(N00C.NE.0.OR.N011.NE.0) KEYCHG=1
  179  CONTINUE
       IF(KEYCHG.EQ.0) RETURN
  189 CONTINUE
      RETURN
      ENTRY UPTRTH(GI,GZ)
C**** WHEN THIS ENTRY POINT IS CALLED IT WILL UPDATE THE TRUTH TABLE OF
C     GATES WHICH ARE SUCCESSORS OF GZ BUT NOT SUCCESSORS OF GI *****
      LEVZ=GLEVEL(GZ)
      DO 300 LEV=2,LEVZ
        VEL=LEVZ-LEV+1
        LGL=LGLIST(VEL)
        DO 290 LG=1,LGL
          GP=HLIST(LG,VEL)
          IF(GP.EQ.GI
     1    .OR. SUC$MX(GI,GP).GE.1
     2    .OR. SUC$MX(GZ,GP).LE.0
     3    .OR. LIPRED(GP).LE.0) GO TO 290
          BSGP=(GP-1)*N2
          DO 210 TH=1,N2
            P$(1,BSGP+TH)=1
            PSUM(TH)=0
  210     CONTINUE
          LIP=LIPRED(GP)
          DO 260 LP=1,LIP
            GR=IPRED(LP,GP)
            BSGR=(GR-1)*N2
            DO 230 TH=1,N2
              PSUM(TH)=PSUM(TH)+P$(1,BSGR+TH)
  230       CONTINUE
  260     CONTINUE
          DO 280 TH=1,N2
            IF(PSUM(TH).GE.1) P$(1,BSGP+TH)=0
  280     CONTINUE
  290   CONTINUE
  300 CONTINUE
      RETURN
      END


      SUBROUTINE SUBNET
      IMPLICIT INTEGER*4(A-T,V-Z,$), REAL(U)
C
C     DEFINITIONS OF 'COMMON' VARIABLES CAN BE FOUND IN MAIN PROGRAM.
C
      COMMON NEPMAX
      COMMON    N             , M             , A               , B
     1     ,    R             , N2            , N1              , NR
     2     ,    NM            , KFLAG         , JFLAG           , COST
     3     ,    LEVM          , NRN2          , NM1             , NN2
      COMMON    ISUCC(40,40)  , LISUCC(40)    , IPRED(40,40)    , LIPRED(40)
     1     ,    INC$MX(40,40) , SUC$MX(40,40) , P$(2,1280)      , UNAME(40)
     2     ,    GLEVEL(40)    , LGLIST(40)    , HLIST(40,40)    , TIME
      COMMON    T             , RTCONN(100)   , S               , RSCONN(100)
      COMMON    IFLAG         ,POINTA         ,ESS1S(40)        ,F$1(32)
     1     ,F$UB1             ,INPTCV(32)     ,LISTC(40)        ,POINTC
```

```
     2      ,LISTL(40)          ,POINTL          ,ORIGIN(40)      ,IPATH(40)
     3      ,POINTR             ,VF$1(32)        ,VF$UB1          ,GSMALL(40,32)
      COMMON    POTAB(200,42),PPOTAB(40)      ,LPOTAB(40)      ,NRPLC(2)
     1      ,RPLC(2,40)         ,IDX0(32)        ,IDX0E(32)       ,IDX1(32)
     2      ,IDX1E(32)          ,SUMP(32)        ,SETT1(32)       ,NOT1
     3      ,SETS1(40)          ,NOS1            ,SETS(40)        ,NOS
     4      ,STS                ,SUMS2(32)       ,SETS2(200)      ,NOS2
     5      ,LIP                ,NODE            ,KEYA            ,KEYB
     6      ,NO0                ,NO1             ,NO1E            ,$GT
     7      ,$LTH               ,$PW             ,$NOE            ,G$$$$$
      COMMON                     NOT1SV          ,NOS1SV          ,LMTS2
      DIMENSION X(40),LX(40,2),OUTO(40)
C
      ENTRY PRESUC
    1 CONTINUE
      DO 10 GI=1,NR
       LS=0
       LP=0
       DO 5 GJ=1,NR
        IF(INC$MX(GI,GJ).EQ.0) GO TO 3
         LS=LS+1
         ISUCC(LS,GI)=GJ
         GO TO 5
    3   IF(INC$MX(GJ,GI).EQ.0) GO TO 5
         LP=LP+1
         IPRED(LP,GI)=GJ
    5  CONTINUE
       LISUCC(GI)=LS
       LIPRED(GI)=LP
   10 CONTINUE
C
      ENTRY SUCCES
      DO 21 GI=1,NR
      DO 21 GJ=1,NR
       SUC$MX(GI,GJ)=0
   21 CONTINUE
      DO 30 GJ=N1,NR
      DO 22 GS=1,NR
        X(GS)=0
   22  CONTINUE
       X(GJ)=1
       LO=1
       LX(1,1)=GJ
       V=1
   23  CONTINUE
       V=1-V
       SWO=1+V
       SW1=2-V
       L1=0
       DO 28 LL=1,LO
        GM=LX(LL,SWO)
        LIP=LIPRED(GM)
        IF(LIP.EQ.0) GO TO 28
        DO 26 LP=1,LIP
         GP=IPRED(LP,GM)
         IF(X(GP).GT.0) GO TO 26
          SUC$MX(GP,GJ)=1
          L1=L1+1
          LX(L1,SW1)=GP
          X(GP)=1
   26    CONTINUE
   28  CONTINUE
```

```
      IF(L1.EQ.0) GO TO 30
      LO=L1
      GO TO 23
   30 CONTINUE
C
C     ENTRY LEVEL
      DO 40 GJ=1,NR
       OUTO(GJ)=LISUCC(GJ)
       GLEVEL(GJ)=-1
   40  CONTINUE
      LEV=0
   45 LEV=LEV+1
      G=0
      DO 50 GJ=1,NR
       IF(OUTO(GJ).GT.0 .OR. GLEVEL(GJ).GT.0) GO TO 50
       G=G+1
       HLIST(G,LEV)=GJ
       GLEVEL(GJ)=LEV
   50 CONTINUE
      IF(G.EQ.0) RETURN
      LGLIST(LEV)=G
      DO 60 GG=1,G
       GJ=HLIST(GG,LEV)
       LIP=LIPRED(GJ)
       IF(LIP.EQ.0) GO TO 60
       DO 55 LP=1,LIP
        GP=IPRED(LP,GJ)
        OUTO(GP)=OUTO(GP)-1
   55  CONTINUE
   60 CONTINUE
      LEVM=LEV
      GO TO 45
C
C
C
      ENTRY PVALUE
      DO 100 L=NN2,NRN2
       P$(1,L)=1
  100 CONTINUE
C
      LEV=LEVM
  110 CONTINUE
      LO=LGLIST(LEV)
      DO 130 L=1,LO
       GI=HLIST(L,LEV)
       LIS=LISUCC(GI)
       BSGI=(GI-1)*N2
       LJTH=0
       DO 115 JTH=1,N2
        IF(P$(1,BSGI+JTH).EQ.0) GO TO 115
        LJTH=LJTH+1
        X(LJTH)=JTH
  115  CONTINUE
       IF(LJTH.EQ.0) GO TO 130
       DO 125 LS=1,LIS
        GS=ISUCC(LS,GI)
        BSGS=(GS-1)*N2
        DO 120 LJ=1,LJTH
         P$(1,X(LJ)+BSGS)=0
  120   CONTINUE
  125  CONTINUE
```

```
130 CONTINUE
    LEV=LEV-1
    IF(LEV.GE.2) GO TO 110
    RETURN
    ENTRY RSTRCT(KEYRST)
    KEYRST=0
    IF(LEVM.GT.LMAX)GO TO 160
    DO 150 GI=N1,NP
     IF(LIPRED(GI).GT.FANIN)GO TO 160
     IF(LISUCC(GI).GT.FANOUT)GO TO 160
150 CONTINUE
    RETURN
160 KEYRST=1
    RETURN
    ENTRY UNNECE
***** THIS ENTRY DISCONNECT ALL GATES FROM WHICH THERE IS NO PATH
     TO OUTPUT GATES *****
    TS=T
    DO 209 GI=NM1,NP
     IF(GLEVEL(GI).EQ.1) GO TO 207
     DO 205 GJ=N1,NM
      IF(SUC$MX(GI,GJ).GT.0) GO TO 209
205  CONTINUE
***** GI IS REDUNDANT *****
207 CONTINUE
     LIP=LIPRED(GI)
     IF(LIP.EQ.0) GO TO 206
     DO 203 LI=1,LIP
      GK=IPRED(LI,GI)
    IF(INC$MX(GK,GI).LE.0) GO TO 203
      T=T+1
      RTCONN(T)=100*GK+GI
      INC$MX(GK,GI)=0
203  CONTINUE
206  LIS=LISUCC(GI)
     IF(LIS.EQ.0) GO TO 209
     DO 204 LI=1,LIS
      GK=ISUCC(LI,GI)
    IF(INC$MX(GI,GK).LE.0) GO TO 204
      T=T+1
      RTCONN(T)=100*GI+GK
      INC$MX(GI,GK)=0
204  CONTINUE
209 CONTINUE
    IF(T.GT.TS) GO TO 1
    RETURN
    END
```

BIBLIOGRAPHIC DATA SHEET	1. Report No. UIUCDCS-R-75-714	2.	3. Recipient's Accession No.

4. Title and Subtitle
PROGRAM MANUAL: NOR NETWORK TRANSDUCTION BY GERERALIZED GATE MERGING AND SUBSTITUTION (Reference Manual of NOR Network Transduction Programs NETTRA-G3 and NETTRA-G4)

5. Report Date
April 1975

6.

7. Author(s)
H.C. LAI

8. Performing Organization Rept. No.

9. Performing Organization Name and Address
Department of Computer Science
University of Illinois at Urbana-Champaign
Urbana, Illinois 61801

10. Project/Task/Work Unit No.

11. Contract/Grant No.
NSF GJ-40221

12. Sponsoring Organization Name and Address
National Science Foundation
1800 G Street, N.W.
Washington, D.C. 20550

13. Type of Report & Period Covered
Technical

14.

15. Supplementary Notes

16. Abstracts
This is a reference manual for NOR network transduction programs NETTRA-G3 and NETTRA-G4. NETTRA-G3 reduces the number of gates in a given network by means of merging of gates whereas NETTRA-G4 reduces the number of gates by means of substitution for all output connections of a selected gate. The principles of these programs will be discussed in more detail in another paper by the author and Y. Kambayashi.

17. Key Words and Document Analysis. 17a. Descriptors

Logic design, logic circuits, logical elements, programs (computers).

17b. Identifiers/Open-Ended Terms
Program manual, computer-aided-design, permissible functions, network transduction, network tranformation, gate merging, gate substitution, NETTRA-G3, NETTRA-G4, NOR, CSPF.

17c. COSATI Field/Group

18. Availability Statement	19. Security Class (This Report)	21. No. of Pages
Release Unlimited	UNCLASSIFIED	107
	20. Security Class (This Page) UNCLASSIFIED	22. Price